НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

Стельмах Олександр Петрович

УДК 004.94:656.056

ДИСЕРТАЦІЯ
МЕТОДИ ТА МОДЕЛІ АНАЛІЗУ ТРАНСПОРТНИХ СИСТЕМ В УМОВАХ НЕСТАЦІОНАРНОСТІ ПАРАМЕТРІВ ТРАНСПОРТНОГО ПОТОКУ

122Комп’ютерні науки
12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело_____

Науковий керівник Стеценко Інна Вячеславівна, д.т.н., професор

Київ – 2021

АНОТАЦІЯ

Стельмах О.П. Методи та моделі аналізу транспортних систем в умовах нестаціонарності параметрів транспортного потоку. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 – Комп'ютерні науки та 12 – Інформаційні технології. – Національний Технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського», Київ, 2021.

Дисертаційна робота присвячена розробці інформаційної технології для підвищення точності визначення інтенсивності транспортного руху на основі аналізу даних відеопотоку в режимі реального часу в умовах нестаціонарності параметрів транспортного потоку.

Висвітлено основні тенденції застосування інформаційних транспортних систем та розглянуто сучасні інформаційні транспортні системи виявлення транспортних засобів.

В даному розділі розглянуто існуючі транспортні системи визначення інтенсивності транспортного руху. Незважаючи на наявність розвинутих технологій керування дорожніх рухом, на сьогоднішній день задача про визначення інтенсивності в Україні та в світі все ще залишається не розв'язаною через ряд недоліків існуючих засобів моніторингу транспортних систем – низька точність, складність реалізації системи та ін. Разом з тим існує величезна потреба у точному визначенні інтенсивності транспортного руху для підвищення якості управління транспортним рухом, що призведе до покращення економіки та екології міст і благополуччя населення.

Розглянуто існуючі транспортні системи прогнозування інтенсивності транспортного руху. Хоча розроблено велику кількість методів, прогнозування транспортного потоку все ще лишається складним завданням і потребує розроблення нових точних методів.

З розвитком сучасних технологій комп'ютерного зору та зі збільшенням кількості відео-камер спостереження за дорожнім рухом з'являється все більше можливостей для побудови нових ІТС для визначення інтенсивності транспортних потоків, створення якої є актуальною задачею для підвищення якості управління транспортним рухом.

Вперше було розроблено метод визначення показника завантаженості смуги транспортного руху, який отримав назву TLCR (Traffic Lane Congestion Ratio), для оцінки заторів на одній смузі. Розроблено алгоритм обробки зображень з метою виявлення транспортних засобів на наявній ділянці. Запропоновано метод визначення інтенсивності дорожнього руху за послідовними значеннями показника завантаженості смуги дорожнього руху за даними відеоряду. Правильно визначені значення параметрів завантаженості смуги транспортного руху TLCR та інтенсивності дорожнього руху дозволить системі управління знайти оптимізовані параметри та уникнути заторів.

Розроблено технологію визначення руху об'єктів у відео потоці на основі модуля «bioinspired». Для цього було використано адаптований клас Retina модуля «bioinspired», в якому знаходяться просторово-часовий фільтр двох інформаційних каналів моделі сітківки ока для детектування руху транспортних засобів. Під час обробки зображення спочатку вказуються координати області зображення, де необхідно виявляти рух, після чого вхідне зображення трансформується з кольорового до відтінків сірого, і далі зображення обробляється в модулі «magnocellular» та перевіряється медіанним фільтром на рівень ентропії та результат обробки порівнюється з пороговим значенням.

Розроблено програмний компонент для визначення інтенсивності дорожнього руху в двох реалізаціях. Експериментально доведено значну перевагу реалізації на основі модуля bioinspired над реалізацією на основі порівняння двох кадрів. Програмний компонент визначення інтенсивності транспортного руху, який розроблений, є складовою частиною інформаційної системи управління транспортним рухом.

Розроблена технологія визначення інтенсивності дорожнього руху з використанням нейронної мережі U-net, яка має високу ефективність та результативність під час сегментації зображення і відома надійністю під час роботи з великими наборами даних. Було використано методи U-net – методи кодування та декодування для злиття базової інформації та інформації високого рівня. Для навчання нейронної мережі було використано алгоритм оптимізації SGD для аналізу градієнта об'єктів. Вхідні зображення розрізались на сегменти розміром 128 на 128 пікселів. Для тренування нейронної мережі U-net було використано набір даних з 10000 зображень.

Експериментально доведено перевагу використання нейронної мережі U-net для задачі визначення інтенсивності руху та показника завантаженості TLCR.

Розроблена технологія визначення інтенсивності дорожнього руху за даними відеоряду, що надходять з відеокамери спостереження. Було вдосконалено алгоритм визначення показника завантаженості транспортної ділянки TLCR надає можливість враховувати тільки автомобілі, які рухаються по досліджуваній смузі. Розроблений метод визначення інтенсивності дорожнього руху на основі послідовних значень показника завантаженості має наступні переваги над іншими подібними системами: швидкість обробки даних, точність, відсутність необхідності додаткового обладнання (наприклад датчики) та низька вартість.

Розроблено алгоритм та інформаційну систему для довгострокового прогнозування показника завантаженості транспортної ділянки TLCR для подальшої оцінки стану дорожнього руху. Інформаційна система прогнозування базується на моделі навчання з рекурентною нейронною мережею LSTM. Розроблена система навчається на тренувальному відео отриманих з камер дорожнього руху записного протягом одного тижня для прогнозування показника завантаженості транспортної ділянки TLCR для кожного дня тижня.

Розроблено алгоритм виявлення дорожніх заторів за показником завантаженості транспортної ділянки TLCR отриманих з зображень отриманих з відеокамер, встановлених у різних місцях міста. Алгоритм дозволяє класифікувати затори за трьома рівнями завантаженості: низька, середня та висока завантаженість. Розроблений алгоритм досліджували на експериментальних даних записів з дорожніх відеокамер та було отримано задовільні результати виявлення та класифікації рівнів завантаженості.

Послідовність обробки та перетворень даних складають нову технологію визначення інтенсивності дорожнього руху, що забезпечує високу точність оцінки інтенсивності руху транспортних засобів на ділянці дорожнього руху. Розроблена технологія може працювати в умовах нестаціонарності параметрів транспортного руху. Завдяки використанню сегментації, замість класифікації, а також специфічного набору даних для навчання, технологія позбавлена таких недоліків як неправильно підібраний ракурс та відсутність транспортного засобу в існуючих базах даних для навчання. Запропонована система успішно підрахувала транспортні засоби з високою точністю, – середні значення F-міра та точність (Accuracy) досягли 0,9967 та 0,9935 відповідно.

Досліджено точність розробленої інформаційної системи довгострокового прогнозування показника завантаженості транспортної ділянки TLCR, яка базується на моделі навчання з рекурентною нейронною мережею LSTM. Отримана експериментальна середня точність 0,914 для п’яти днів (два вихідних та три робочих дні) демонструє високу ефективність результатів прогнозування.

Ключові слова: аналіз зображень; розпізнавання образів; OpenCV; дорожній рух; інтенсивність транспортного руху; показник завантаженості транспортного руху; TLCR; нейронна мережа; U-net; інформаційна система управління.

Список публікацій здобувача

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. Stetsenko, I.V., Stelmakh, O. (2020). Traffic Lane Congestion Ratio Evaluation by Video Data. Advances in Intelligent Systems and Computing, 1019, 172-181. Springer, Cham. ISSN 2194-5357.https://doi.org/10.1007/978-3-030-25741-5_18 (Scopus)

2. Stelmakh O.P., Stetsenko I.V., Velyhotskyi D.V. (2020). Information technology of video data processing for traffic intensity monitoring. *Control systems & computers*, 3 (287), 49-59.

3. Стеценко І.В., Стельмах О.П. (2020). Технологія визначення інтенсивності дорожнього руху за даними відеоряду. *Технічні науки та технології: науковий журнал*, 2, 116-125.

4. Стеценко І.В., Стельмах О.П. (2017). Програмний компонент визначення інтенсивності транспортних потоків. *Вісник Національного технічного університету України «КПІ імені І. Сікорського»*, 66, 94-100.

ABSTRACT

Stelmakh O.P. Methods and models of analysis of transport systems in the conditions of non-stationary parameters of transport flow. - Qualified scientific work on the rights of the manuscript.

Dissertation for the degree of Doctor of Philosophy in the specialty 122 - Computer Science and 12 - Information Technology. - National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, 2021.

The dissertation work is devoted to the development of information technology to increase the accuracy of determining the intensity of traffic based on the analysis of video data in real time in conditions of non-stationary parameters of traffic flow.

The basic tendencies of application of information transport systems are covered and modern information transport systems of detection of vehicles are considered.

This section discusses the existing transport systems for determining the intensity of traffic. Despite the availability of advanced traffic management technologies, today the task of determining the intensity in Ukraine and in the world still remains unsolved due to a number of shortcomings of existing means of monitoring transport systems - low accuracy, complexity of the system, etc. However, there is a great need to accurately determine the intensity of traffic to improve the quality of traffic management, which will improve the economy and ecology of cities and the well-being of the population.

Existing transport systems for forecasting traffic intensity are considered. Although a large number of methods have been developed, traffic flow forecasting is still a challenge and requires the development of new accurate methods.

With the development of modern computer vision technology and the increasing number of video surveillance cameras, there are more and more opportunities to build new ITS to determine the intensity of traffic, the creation of which is an urgent task to improve the quality of traffic management.

For the first time, a method for determining the congestion of the traffic lane, called TLCR (Traffic Lane Congestion Ratio), was developed to estimate congestion

in one lane. An algorithm for image processing to identify vehicles on the existing site has been developed. A method for determining the traffic intensity by successive values of the traffic lane congestion index according to the video series is proposed. Correctly defined values of the TLCR lane load and traffic intensity parameters will allow the control system to find optimized parameters and avoid congestion.

The technology of determining the movement of objects in the video stream based on the "bioinspired" module has been developed. For this purpose, an adapted Retina class of the "bioinspired" module was used, which contains a space-time filter of two information channels of the retina model for detecting the movement of vehicles. During image processing, the coordinates of the image area where the motion is to be detected are first specified, after which the input image is transformed from color to grayscale, and then the image is processed in the magnocellular module and checked by the median filter for entropy and the processing result is compared.

A software component for determining traffic intensity in two implementations has been developed. The significant advantage of the implementation based on the bioinspired module over the implementation based on the comparison of two frames has been experimentally proved. The software component for determining the intensity of traffic, which is developed, is an integral part of the information system of traffic management.

The technology of determining the intensity of traffic using the neural network U-net, which has high efficiency and effectiveness in image segmentation and is known for reliability when working with large data sets. U-net methods were used - encoding and decoding methods for merging basic information and high-level information. To train the neural network, the SGD optimization algorithm was used to analyze the gradient of objects. The input images were cut into 128 by 128 pixel segments. A data set of 10,000 images was used to train the U-net neural network.

The advantage of using the U-net neural network for the problem of determining the traffic intensity and TLCR load index has been experimentally proved.

The technology of determining the intensity of traffic based on video data coming from a video surveillance camera has been developed. The algorithm for determining the load index of the transport section of the TLCR has been improved, making it possible to take into account only cars moving in the studied lane. The developed method of determining traffic intensity based on consistent values of congestion has the following advantages over other similar systems: data processing speed, accuracy, no need for additional equipment (eg sensors) and low cost.

An algorithm and information system for long-term forecasting of the TLCR traffic congestion index for further assessment of the traffic condition have been developed. The forecasting information system is based on a learning model with a recurrent LSTM neural network. The developed system is trained on a training video obtained from traffic cameras recorded for one week to predict the load of the TLCR for each day of the week.

An algorithm for detecting traffic jams based on the load index of the TLCR transport section obtained from images obtained from video cameras installed in different parts of the city has been developed. The algorithm allows to classify traffic jams according to three levels of congestion: low, medium and high congestion. The developed algorithm was investigated on the basis of experimental data of recordings from road video cameras and satisfactory results of detection and classification of load levels were obtained.

The sequence of data processing and transformations constitute a new technology for determining the intensity of traffic, which provides high accuracy in estimating the intensity of traffic on the road. The developed technology can work in conditions of non-stationary traffic parameters. Due to the use of segmentation instead of classification, as well as a specific set of data for training, the technology is devoid of such shortcomings as incorrectly selected angle and the lack of a vehicle in existing databases for training. The proposed system successfully calculated vehicles with high accuracy - the average values of F-measure and accuracy (Accuracy) reached 0.9967 and 0.9935, respectively.

The accuracy of the developed information system of long-term forecasting of the load index of the transport section TLCR, which is based on the training model with a recurrent neural network LSTM, is investigated. The obtained experimental average accuracy of 0.914 for five days (two days off and three working days) demonstrates high efficiency of forecasting results.

Keywords: image analysis; pattern recognition; OpenCV; Road traffic; traffic intensity; traffic congestion index; TLCR; neural network; U-net; management information system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

OpenCV – бібліотека програмних засобів для обробки та аналізу вмісту зображень

LSTM (Long Short-Term Memory) – тип рекурентної нейронної мережі

TLCR (Traffic Lane Congestion Ratio) – коефіцієнт завантаженості смуги транспортного руху

U-net – тип кодера-декодера мережевої архітектури для сегментації зображень

ВСТУП

Актуальність теми. Затори на дорогах є величезною проблемою для всіх учасників дорожнього руху і причиною їх є зростаюча інтенсивність руху, низька пропускна спроможність транспортних шляхів та, водночас, незадовільна якість систем управління транспортним рухом. Інтенсивність руху є найважливішим фактором, що впливає на безпеку дорожнього руху. Його значення використовується при плануванні і проведенні дорожньо-будівельних робіт на автомобільних дорогах, розробці планів і заходів з розвитку дорожньої мережі, визначенні обсягу інвестицій в дорожню галузь [Pechatnova E. Mathematical modeling of changes daily traffic volume. The Russian Automobile and Highway Industry Journal. 2017. P. 145-151.].

Посібник з проектування доріг вказує на 10% темп зростання інтенсивності руху для всіх національних автомобільних доріг щорічно [Hoque Md. Sh., Ullah M. A., Nikraz H. Investigation of traffic flow characteristics of Dhaka-Syijiet highway (N-2) of Bangladesh. International journal of civil engineering and technology (IJCET).2013. Vol. 4. Issue 4. P. 55-65.]. Відповідність цього значення фактичному значенню темпу зростання може бути оцінена лише у випадку, якщо аналіз темпу зростання виконується на основі фактичних даних про постійний потік трафіку, оскільки зростання інтенсивності трафіку не є рівномірним на ділянках транспортного руху. Тому вимога до даних про безперебійний трафік та їх належного аналізу для досягнення автентичних характеристик потоку руху є необхідною. Правильно визначені характеристики транспортних потоків, такі як інтенсивність та завантаженість, допоможуть визначити вказівки майбутнього управління дорожнім рухом, які є обов'язковими для ефективного управління транспортними потоками.

Тому одним з найважливіших завдань є збір даних, оскільки від правильності визначення показників дорожнього руху залежить якість роботи системи в цілому. Нинішній прогрес в області комп'ютерного зору дозволяє по новому поглянути на існуючу проблему.

Зв’язок роботи з науковими програмами, планами, темами.

Дисертаційна робота тісно пов'язана з науковими розробками, що здійснюються на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського». Тема дисертації відповідає науковому напряму «Інтелектуальні інформаційні та інформаційно-аналітичні технології. Інтегровані системи баз даних та знань. Національні інформаційні ресурси» переліку пріоритетних тематичних напрямів наукових досліджень і науково-технічних розробок на період до 2020 року, затвердженого постановою Кабінету Міністрів України №942 від 07.09.2011р. У дисертації запропоновані методи та моделі, які спрямовані на вирішення завдань, поставлених у Міській цільовій програмі розвитку транспортної інфраструктури м. Києва на 2019-2023 роки у п.10 «Зміна моделі управління транспортної інфраструктури міста» та п.4 «Інформатизація транспортної системи», а саме на вдосконалення системи моніторингу транспортної системи міста та впровадження інтелектуальних систем керування дорожнім рухом.

Мета і завдання дослідження. Метою дисертаційної роботи є підвищення точності визначення інтенсивності транспортного руху на основі аналізу даних відеопотоку в режимі реального часу.

Для досягнення вказаної мети потрібно вирішити такі завдання:

- розробити метод визначення показника завантаженості;
- розробити алгоритм обробки зображень з метою виявлення транспорту на наявній ділянці;
- розробити технологію визначення інтенсивності дорожнього руху за даними відеоряду;
- розробити метод визначення інтенсивності за послідовними значеннями показника завантаженості смуги дорожнього руху;
- розробити метод прогнозування інтенсивності.

Об’єкт дослідження – процес визначення інтенсивності транспортного руху на основі аналізу даних відеопотоку.

Предмет дослідження – моделі та методи аналізу даних відеопотоку з метою визначення інтенсивності транспортного руху.

Методи дослідження. Методологічною основою дослідження є системне опрацювання та аналіз теоретичного матеріалу, присвяченого визначенню інтенсивності транспортного руху на основі аналізу даних відеопотоку та пошуку шляхів підвищення такої інформаційної технології. В процесі даного дослідження були використані методи теорії сигналів та обробки зображень, статистичного аналізу, а також методи машинного навчання, зокрема синтезу та моделювання систем нечіткого логічного висновку.

Наукова новизна отриманих результатів:

- *Вперше* запропонований показник завантаженості смуги дорожнього руху TLCR, що надає можливість на основі обробки кадру відеоряду оцінити зайнятість ділянки транспортного руху транспортними засобами і який, на відміну від існуючого підходу оцінювання кількості транспортних засобів, містить опосередковано інформацію не тільки про кількість транспортних засобів, але і про їх розміри, а також про наближеність ситуації на ділянці до затору.
- *Вперше* розроблений метод визначення показника інтенсивності дорожнього руху на основі математичної моделі, що описує зв’язок між показником інтенсивності дорожнього руху та послідовними значеннями показника завантаженості смуги дорожнього руху TLCR.
- *Вперше* розроблена інформаційна технологія визначення інтенсивності дорожнього руху на основі обробки даних відеоряду, що більш точно оцінює показник інтенсивності у порівнянні з існуючими і надає можливість автоматизувати моніторинг та прогнозування інтенсивності транспортного руху.
- *Набув подальшого розвитку* метод детекції транспортних засобів у відеопотоці з використанням нейромережі U-Net за рахунок унікальних даних для навчання та меншої кількості оброблюваних зображень

Практичне значення отриманих результатів. Одержані результати дозволяють оцінювати стан інтенсивності руху, що дозволило реалізувати інформаційну технологію аналізу транспортних систем в умовах нестаціонарності параметрів транспортного потоку. Розроблено програмний комплекс, що дозволяє використовувати будь яке апаратне забезпечення (камери), налаштовувати положення смуг руху, збирати дані для навчання мереж, визначати та прогнозувати характеристики транспортних потоків на будь яких ділянках руху.

Особистий внесок здобувача. Дисертація є результатом самостійних наукових досліджень, в яких вкладено авторський підхід до побудови інформаційної технології аналізу транспортних систем в умовах нестаціонарності параметрів транспортного потоку. Наукові положення та основні результати, які містяться в дисертації, отримані здобувачем самостійно у процесі науково-дослідницької роботи. В роботах, опублікованих у співавторстві, дисертанту належать: [1] – обґрунтовано використання запропонованого показника завантаженості TLCR; [2] – розроблена технологія визначення інтенсивності дорожнього руху за даними відеоряду; [3] – вдосконалений алгоритм визначення показника завантаженості транспортної дільниці TLCR дозволяє враховувати тільки автомобілі, які рухаються по досліджуваної смузі.; [4] – розроблено програмний компонент для визначення інтенсивності дорожнього руху за допомогою методів OpenCV.

Апробація результатів дисертації. Основні результати роботи опубліковано та обговорювались на міжнародних та всеукраїнських наукових конференціях, зокрема на: XIV міжнародній науково-практичній конференції “Mathematical Modeling and Simulation of Systems”, MODS 2019 (м. Чернігів, 2019 р.).

Публікації. За результатами дисертаційних досліджень опубліковано 4 наукові праці, в тому числі 3 статті у наукових фахових виданнях України. Публікації входять до наступних наукометричних баз даних з міжнародним індексом цитування: Scopus – 1.

Структура і обсяг роботи. Дисертаційна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел із 128 найменувань та додатків. Загальний обсяг дисертації становить 145 сторінок, з яких 99 сторінок основного тексту, 3 додатки на 32 сторінках, та містить 61 рисунок, 13 формул, 11 таблиць.

РОЗДІЛ 1
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ВИЗНАЧЕННЯ ІНТЕНСИВНОСТІ ТРАНСПОРТНОГО РУХУ ЗА ДАНИМИ ВІДЕОПОТОКУ

Щороку на дорогах стає все більше автомобілів, що призводить до збільшення інтенсивності руху. Однією з причин заторів у містах є те, що дороги не спроектовані для постійно зростаючої кількості транспортних засобів. Крім того, розширення вулиці, особливо в місцях перехрестя, вимагає величезних фінансових витрат з міського бюджету. Величезна нестача дорожньої мережі міста призводить до заторів, якщо трапиться аварія чи ремонт дороги, а водій не ризикує застрягти лише вночі. Через обмежений простір для будівництва нової інфраструктури, включаючи обмежену кількість ресурсів, єдиним способом вирішення проблеми підвищення якості трафіку є вдосконалення системи управління. Система повинна керувати параметрами транспортних потоків як інтенсивністю руху або іншим показником, який показує поточну кількість транспорту на дорогах або інші умови (автокатастрофа, ремонтні роботи).

1.1 Інформаційні транспортні системи

Інформаційні транспортні системи (ITC) [1] призначені для нагляду за дорожнім рухом збирають та аналізують дані автомобільних перевезень з метою покращення дорожнього руху та безпеки. Оскільки транспортні засоби складають головну складову автомобільних перевезень, необхідно виміряти їх відповідні параметри, такі як потік, швидкість, напрямок та щільність. Ранні методи вимірювання, в основному, були зосереджені на фізичних вимірах, наприклад, радіолокаційних, лазерних, індукційних котушках або потрійних контурах, які, певною мірою, потребували вдосконаленого обладнання [2-5]. Завдяки досягненню технологій камер та обробки зображень було показано, що

ці вимірювання можна проводити також ефективно і з використанням відео-камер [6, 7].

У роботі [8, 9] було введено трикадровий різницевий метод, який обчислює контур транспортного засобу в трьох послідовних кадрах, тоді як у [9] алгоритм оптичного потоку Горна–Шунка використовується для обчислення зміщення виявленого контуру, а пізніше - для визначення швидкості руху транспортного засобу. Силует рухомого об’єкта також можна розглянути для отримання більш точного вимірювання швидкості [10, 11]. Ці методи витягують дискримінаційні ознаки безпосередньо з усього транспортного засобу на різній висоті, а не з конкретної горизонтальної площини.

Затори є однією з найбільш нагальних проблем управління дорожнім рухом у всьому світі, особливо для країн, що розвиваються. В роботі [12] описана розроблена система управління та контролю дорожнього руху в міських районах - відеоінтелектуальна транспортна система (MoVITS), що містить набір функціональних можливостей, включаючи виявлення та відстеження транспортних засобів, розпізнавання транспортних засобів, оцінку швидкості, виявлення аномалій та аналіз потоку руху.

Класифікація типу транспортного засобу [12] відіграє дуже важливу роль в ITC, оскільки вона може бути використана для ідентифікації типу транспортного засобу у випадку аномалій, підрахунку транспортних засобів за типом, виявлення дорожніх порушень для деяких конкретних типів тощо. Це дрібнозерниста класифікація. На відміну від типових задач класифікації, заснованих на класифікації об'єктів з різних категорій, дрібнодисперсна класифікаційна робота полягає у виділенні об'єктів однієї категорії. Цей тип класифікації є складним завданням навіть для людей. Ця складність зумовлена не лише загальними характеристиками між деякими класами транспортних засобів, а й впливом кута огляду.

Нещодавні розробки ITC та графічних процесорів (GPU) призвели до того, що велика увага приділялася автоматичному розпізнаванню номерних знаків автомобіля (VLPR) у кількох областях досліджень [13]. LPR вважається

дуже значущим у різних сферах застосування, таких як безпілотні стоянки, управління безпекою регіонів без нагляду, а також управління безпекою руху [14]. На жаль, ці операції виснажливі за рахунок чіткого формату табличок та обмежень динамічного зовнішнього освітлення, а саме фону, яскравості, швидкості транспортного засобу та відстані між камерою та транспортними засобами на момент отримання зображення. Отже, багато методів можна реалізувати з обмеженими правилами, такими як постійне освітлення, низька швидкість транспортного засобу, виділені шляхи та статичний фон.

Для вирішення проблеми виявлення об'єктів транспортного засобу, які перекривають один одного при відеоспостереженні дорожнього руху, в роботі [15] запропоновано вдосконалений метод виявлення на основі YOLOv3 (рис.1.1-1.2). Для покращення згорткового шару YOLOv3 була використана техніка інвертованих залишків та модулі об'єднання просторових пірамід (SPP) (рис.1.3). А для того, щоб впоратися з перекриттям транспортних засобів у відео дорожнього руху (рис.1.4-1.5) було використано придушення не-максимумів (Soft-NMS), яке означає, що пікселями границь оголошуються точки, в яких досягається локальний максимум градієнта в напрямку вектора градієнта.

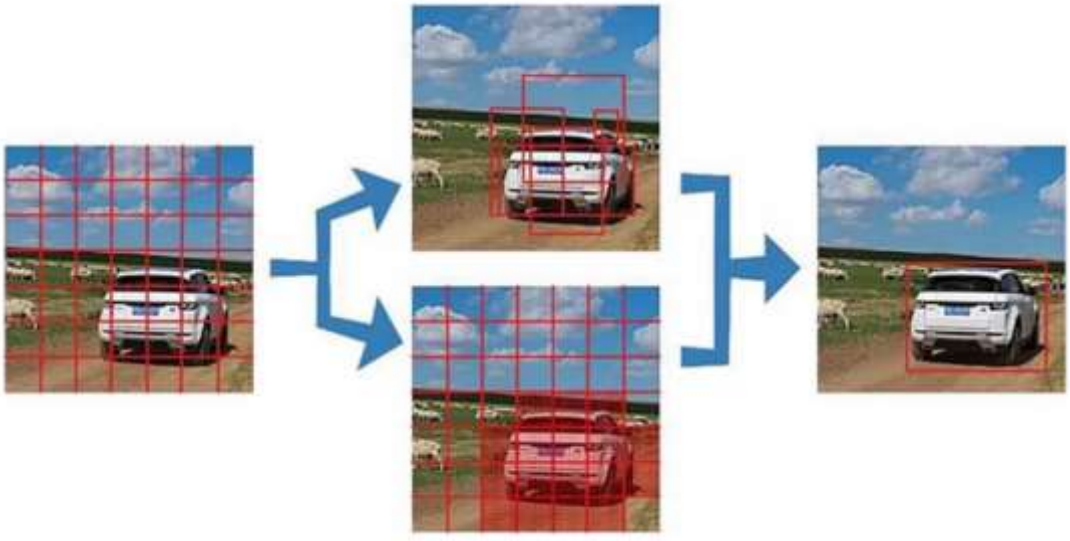


Рис. 1.1. Метод виявлення YOLO [15]

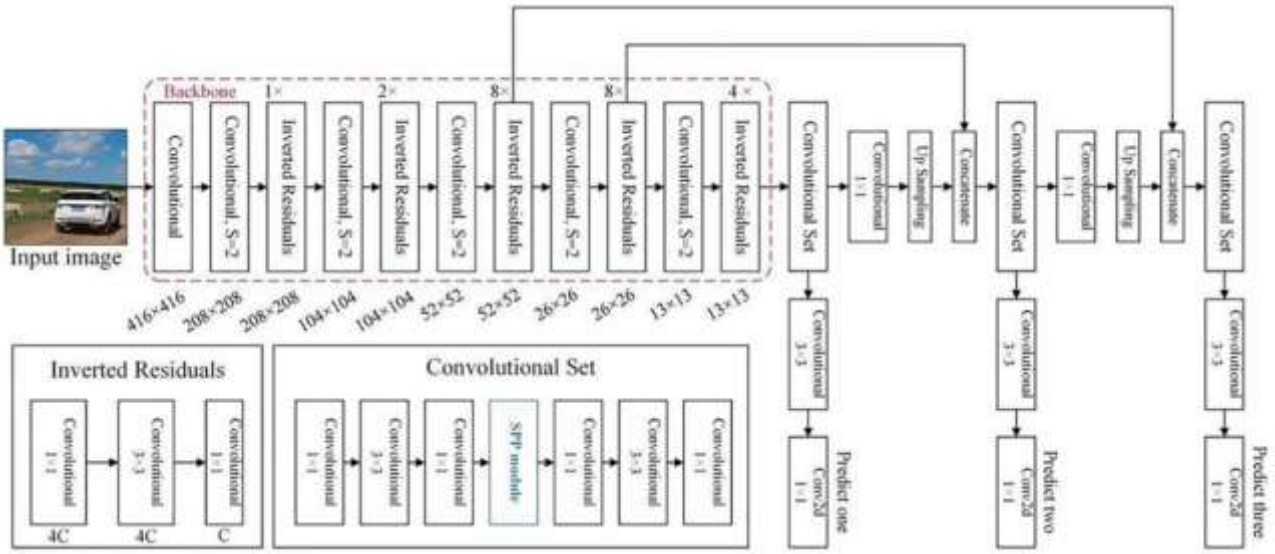


Рис. 1.2. Модель виявлення транспортного засобу. Модель розділена на дві частини, магістральну мережу та мережу виявлення [15]

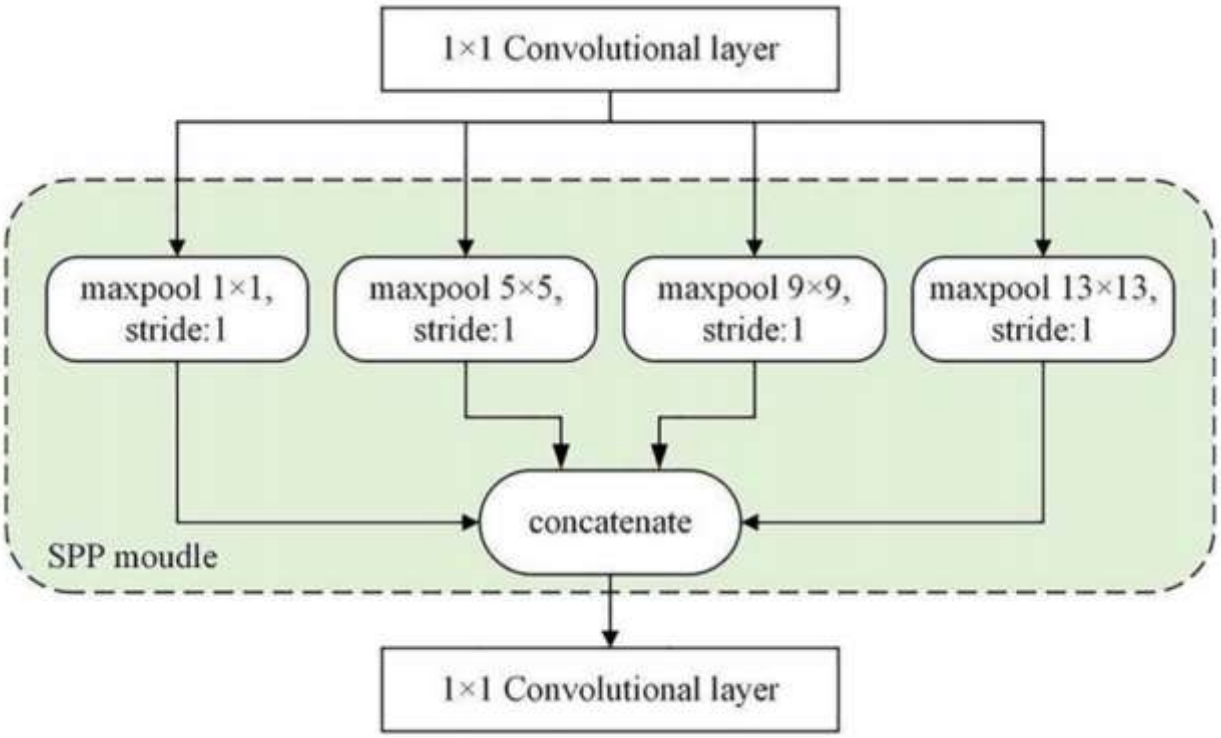


Рис. 1.3. Огляд моделі SPP [15]

В роботі [16] порівнювались чотири методи глибокого навчання для виявлення об’єктів на зображенні: метод Locality Sensitive (LSM), повністю згорнуті мережі (FCNN), YOLO та U-net. Саме U-net продемонстрував кращі результати при сегментації об’єктів на зображенні.

Рис. 1.4. Зразки позначених зображень [15]

Рис. 1.5. Приклади виявлення транспортних засобів [15]

Погодні явища природи, такі як туман, впливають на точність сегментації зображення дороги та виявлення дорожніх об’єктів. В роботі [17] пропонуються методи видалення туману з зображення для покрашення точності виявлення дорожніх об’єктів.

В роботі [18] запропоновано новий підхід до виявлення місць заторів на основі групування траєкторій перебування на місці з використанням GPS-трекерів. В роботі [19] запропоновано метод обробки зображення доріг для

віддаленого зондування для отримання автоматично актуальної та точної інформації про дороги.

Ройовий інтелект [20] - дуже перспективна парадигма для роботи з такими складними та динамічними системами для побудови інтелектуальних міських додатків. Для компактного архівування та ефективної передачі величезної кількості відеоспостереження у роботі [21] запропоновано підхід гібридного стиснення відео за допомогою компенсації руху переднього плану. У роботі [22] використали енергетичним фільтр Габора та шаблон змінної шкали з функціями злиття (CFGVF) для зменшення проблеми впливу оточуючого фону на алгоритм кореляційного фільтра (CF) для візуального відстеження.

В роботі [23] запропоновано використання безпілотних літальних апаратів (БПЛА) в поєднанні з технологією штучного інтелекту (ШІ) для розпізнавання заторів (рис. 1.6).

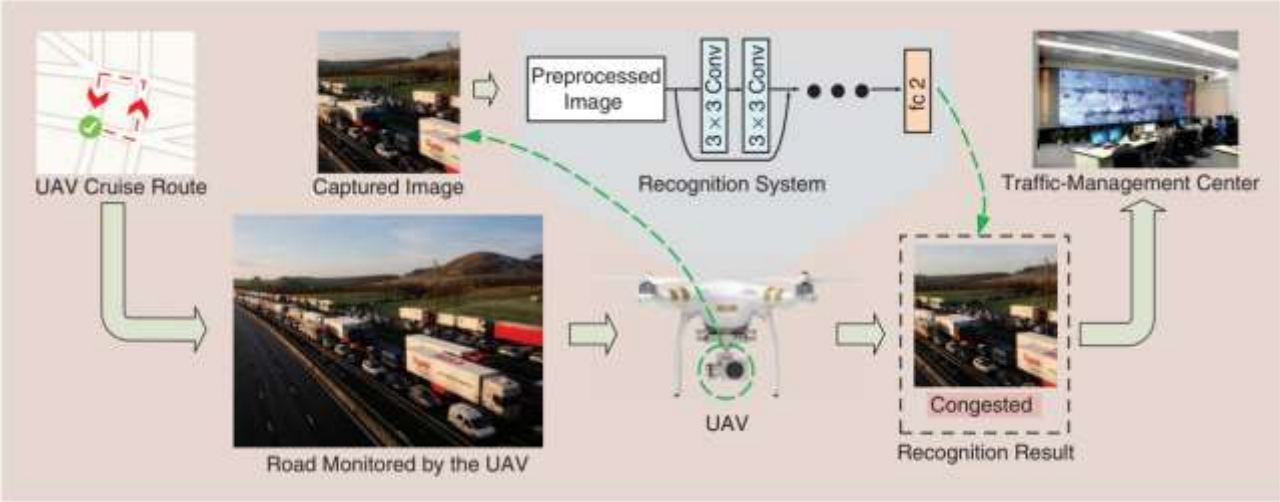


Рис. 1.6.Схема системи моніторингу заторів на дорогах на основі БПЛА [23]

За допомогою системи БПЛА фіксуються зображення дорожніх подій на основі технології планування маршруту в режимі реального часу (рис. 1.7). Після цього аерофотознімки додатково обробляються за допомогою згорткових

нейронних мереж (CNN) (рис. 1.8), результати яких далі передається до центру управління дорожнім рухом [24-26].

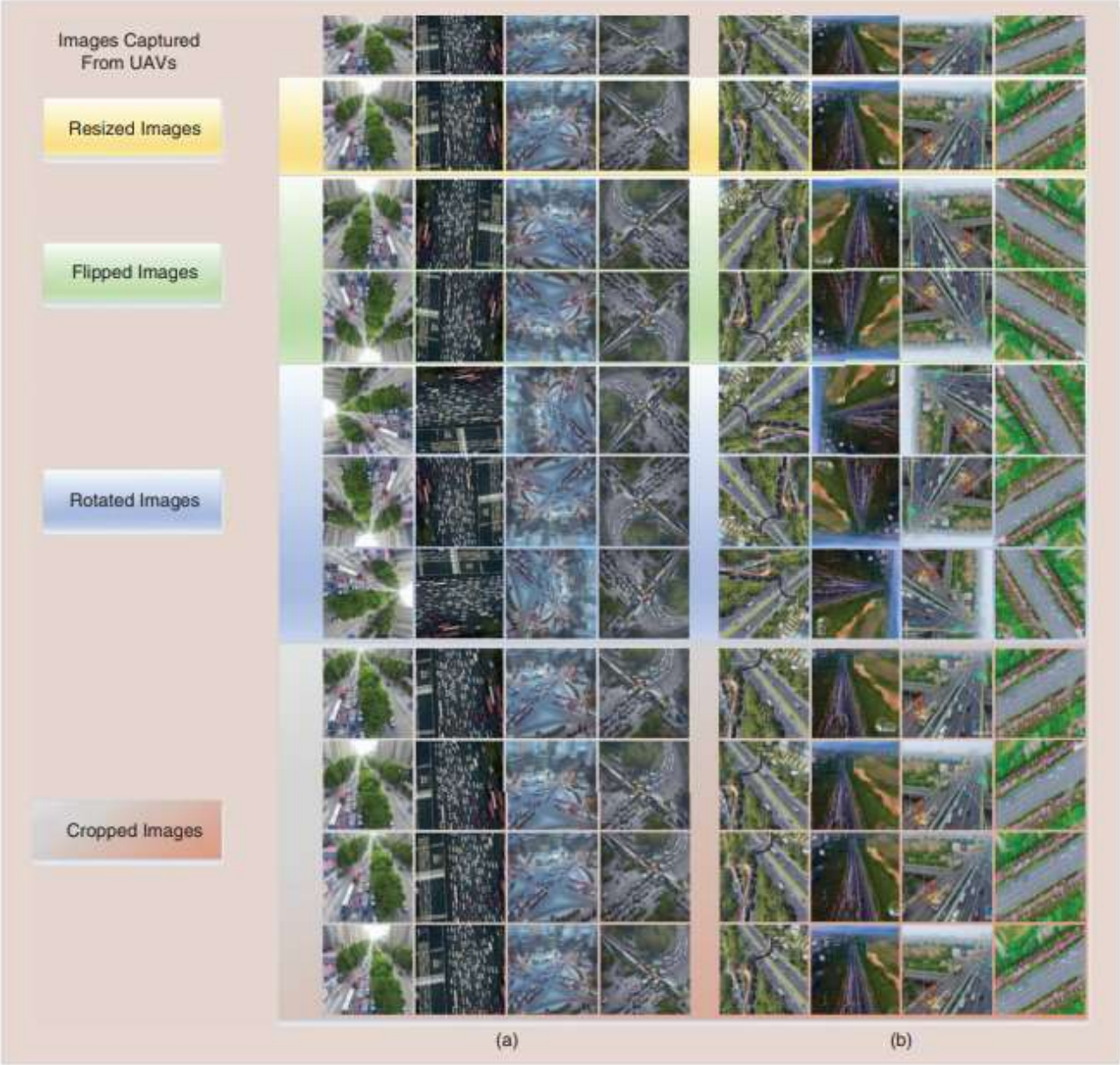


Рис. 1.7. Зображення отримані за допомогою БПЛА з подальшою обробкою за допомогою CNN [23]

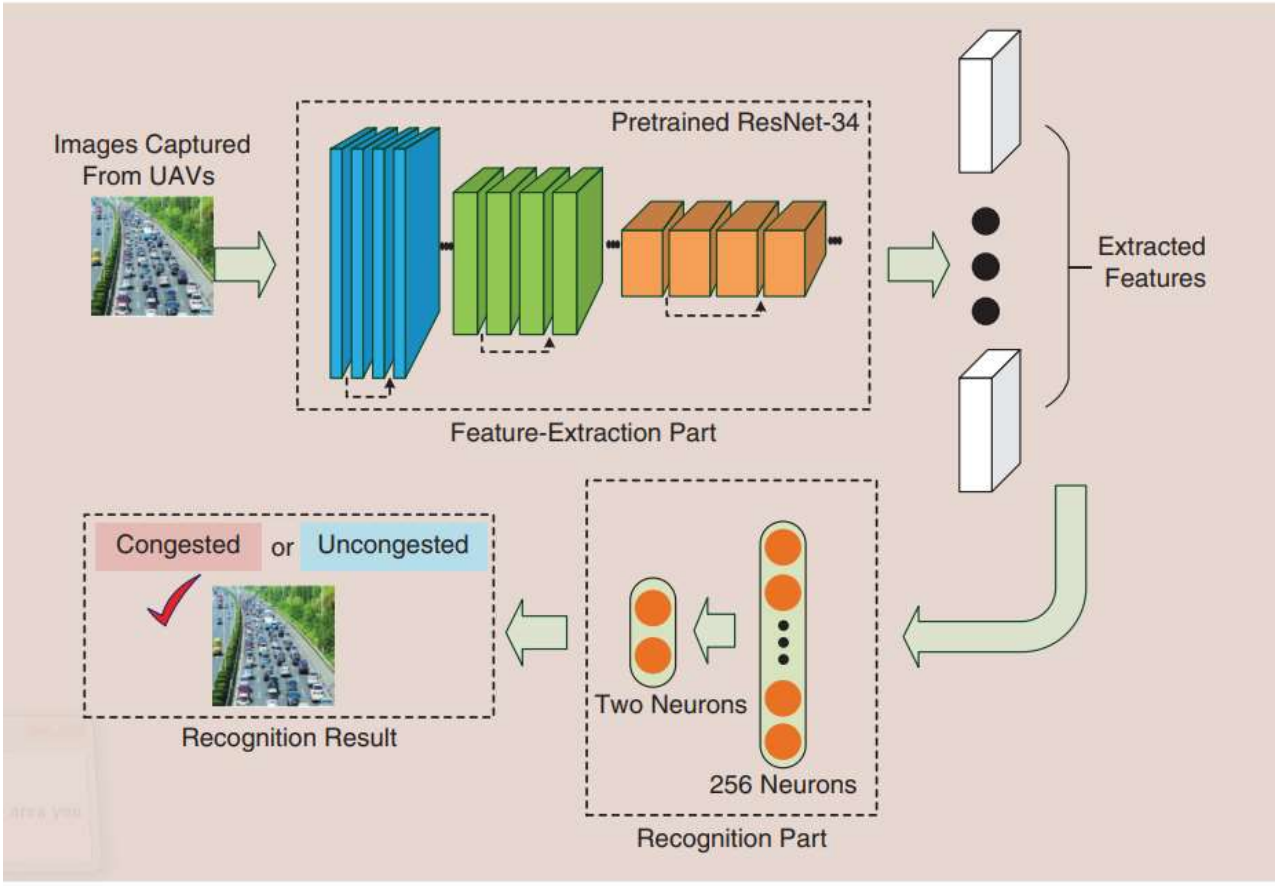


Рис. 1.8. Схема системи розпізнавання на базі CNN [23]

1.2 Транспортні системи виявлення транспортних засобів

В останні роки інтерес до автономних транспортних засобів стрімко зростає. Традиційні транспортні засоби можна вдосконалити завдяки досягненням у галузі штучного інтелекту та його застосуванні в різних галузях. Автовиробники розробляють самокеровані транспортні засоби та продають машини з пристроями безпечного водіння для підтримки водія. Для безпечної автономної їзди необхідне поєднання різних технологій разом із IT-технологіями транспортних засобів [27-30]. Для IT-технологій транспортних засобів використовують наступні датчики: лазерні, радіолокаційні, ультразвукові, лідари, датчики із зарядним пристроєм (ПЗЗ) та додаткові пристрої для зв'язку між транспортними засобами. Однак датчики для безпечного водіння автомобілів залишаються дорогими. Додатковою проблемою є те, що датчики можуть бути легко пошкоджені при незначних

зіткненнях [31, 32]. Більше того, датчики, які збирають тривимірну інформацію (наприклад, лідар), не є практичними для загального користування, оскільки вартість такого датчика співставна з ціною автомобіля [29, 33]. Тому ультразвукові датчики використовуються для збору двовимірної інформації для розпізнавання поруч розташованих об’єктів як допоміжного пристрою для безпечного водіння. Однак більшість цих датчиків розташовані на зовнішній стороні автомобіля (передній та задній бампер та дзеркало заднього виду). У деяких випадках датчики можуть вийти з ладу через пошкодження, спричинені зовнішнім зіткненням або наявністю пилу. Останніми роками більшість транспортних засобів оснащуються пристроями з чорною скринькою для запису умов руху [32, 34-36]. У деяких країнах чорні ящики є обов’язковими для комерційних автомобілів або встановлюються за державної підтримки [32].

В даний час пристрої чорних ящиків для транспортних засобів використовуються для відновлення інформації лише в особливих ситуаціях, таких як транспортні аварії. Тому потрібно було розробити чорний ящик для транспортного засобу, оснащеного функцією автоматичного розпізнавання ситуації водіння заздалегідь та автоматичного надання відповідної інформації для водія для забезпечення безпечного водіння. Деякі з існуючих пристроїв відеозапису включають функцію допомоги водієві (наприклад, виїзд із смуги руху та керівництво виїзду на передній стороні) [37-39]. Однак він має дуже обмежену здатність підтримувати безпеку водіння, оскільки не може виявити та оцінити відстань об'єктів 50 м і далі у всіх напрямках.

Багато досліджень на основі датчиків зору було проведено для виявлення транспортного засобу та симетричного оцінювання відстані [40-43]. Обробку зображень можна застосовувати різними способами на основі особливостей датчика зору. Виявивши транспортний засіб на вхідному зображенні за допомогою обробки зображень, можна отримати різну інформацію про виявлений транспортний засіб. Дослідження виявлення транспортних засобів включають, наприклад, засновані на характеристиках методи узгодження шаблонів [44-48], нейронні мережі або опорні векторні машини [49-51] або

методи на основі форми та руху [45, 52, 53]. Методи виявлення транспортного засобу в основному базуються на особливостях, які приймають незмінну та офіційну форму автомобіля. Дослідження оцінки відстані автомобіля до транспортного засобу включає метод оцінки відстані на основі розміру виявленого автомобіля [41, 43, 54], метод на основі стереокамери [42, 55] та метод, який порівнює розміри дорожньої інфраструктури (смуга руху, напрямна рейка тощо). Методи оцінки відстані автомобіля в основному базуються на виявленій інформації про форму транспортного засобу [38]. Хоча оцінка відстані на основі стереокамери дає відносно точний результат, стереокамера дорожча, а пропускна здатність обчислень обмежена, ніж для монокамери.

В роботі [56] запропоновано метод виявлення транспортного засобу, що рухається вперед, за допомогою однієї камери та оцінки відстані поточного автомобіля від виявленого автомобіля. Щоб зменшити обчислювальні витрати та забезпечити симетричну обробку в режимі реального часу, вхідне зображення дороги було розкладено на зображення з роздільною здатністю, а виявлення автомобіля було виконано для зображення з низькою роздільною здатністю. Запропонований метод використовував каскадний класифікатор із використанням агрегованих характеристик каналів (ACF) та алгоритму AdaBoost [57, 58] для вивчення зображень автомобіля та генерації детекторів транспортних засобів. Алгоритм AdaBoost відомий як класифікатор, який може добре класифікувати два класи за допомогою адаптивного підсилення. Алгоритм AdaBoost - це вдосконалений метод використання ідеї підвищення алгоритму в реальному аналізі даних. Він створює сильний класифікатор із комбінацією декількох тижневих класифікаторів та значень ваги. Алгоритм Adaboost повторно регулює вагу вибіркових даних на початку навчання; дані однаково зважені, але неправильно класифіковані дані збільшують різницю ваги, а добре класифіковані дані зменшують вагу. Класифікатор з найменшим зваженим значенням помилки на кожному етапі є одним слабким класифікатором. Алгоритм Віоли-Джонса [59] відомий як ефективний алгоритм

виявлення об'єктів у реальному часі, що використовує алгоритм AdaBoost із використанням квадратних ознак як слабкого класифікатора.

У запропонованому способі ACF використовуються для зменшення вибірки зображення з великою роздільною здатністю без зменшення інформації про особливості, включеної у вхідне зображення. ACF - це метод, який зменшує вихідний розмір зображення, зменшуючи його, при цьому зберігаючи унікальні особливості зображення та пропускаючи інші функції. Таким чином, ACF прагне зменшити розміри зображення та одночасно зберегти унікальні особливості. Врешті-решт, двовимірна інформація про об’єкт витягується і зменшується до вибірки k разів, щоб генерувати k-ту інформацію про об’єкт при збереженні (k−1)-ої інформації про об’єкт. Це має перевагу у зменшенні обсягу обчислень [60]. У запропонованому методі використовуються алгоритми ACF та AdaBoost [59, 61] для попереднього вивчення області автомобіля для виявлення транспортного засобу. Причиною використання алгоритму AdaBoost є те, що він приблизно в 15 разів швидший, ніж інші алгоритми з використанням нейронних мереж або SVM (підтримка векторної машини), і має високу точність [59].

Згодом було застосовано зворотне перспективне перетворення (IPM) для оцінки відстані від виявленої області транспортного засобу на вхідному зображенні [62, 63]. Інверсна перспективна трансформація - це метод створення зображення з висоти птаха (або виду зверху) шляхом проектування двовимірного зображення, отриманого камерою, у тривимірний простір реального світу та повторного відображення його у двовимірний простір. Хоча неможливо точно розрахувати відстань за допомогою обробки зображень, експерименти показали, що похибка відстані була в межах ± 5 м для рухомих транспортних засобів. Щоб застосувати запропонований метод до системи підтримки безпечного водіння транспортного засобу в дорожньому транспортному середовищі, необхідна обробка в режимі реального часу. У нашому методі розмір зображення був зменшений, зберігаючи інформацію про особливості вхідного зображення, а потім автомобіль було виявлено за

допомогою детектора автомобіля на базі AdaBoost. Потім, використовуючи зворотне перспективне перетворення, двовимірне зображення дороги було нанесене на тривимірний простір, який відповідав реальному простору, а відстань до виявленого транспортного засобу оцінювали симетрично.

Кім та ін. [39] використовував Хаар-подібні характеристики та інформацію про орієнтацію краю для виявлення транспортних засобів, а вони використовували виявлену ширину та інформацію про місцезнаходження для оцінки відстані між транспортними засобами. Для побудови класифікатора, який може виявляти транспортні засоби, для навчання потрібна велика кількість зображень транспортних засобів. Щоб подолати цю складність, було використано шаблон задньої частини транспортного засобу на дорозі для виявлення транспортного засобу. Однак, якщо поверхня транспортного засобу відбиває сонячне світло або якщо на зображенні дороги включені предмети, не пов’язані з транспортним засобом (наприклад, забруднення дороги або смуги руху), може статися неправильне виявлення. Щоб використовувати ширину виявленого транспортного засобу для оцінки відстані транспортного засобу, потрібно точно виявити форму транспортного засобу.

Джеонг та ін. [64] запропонували метод виявлення транспортного засобу та оцінки відстані для запобігання зіткненню. Вони запропонували метод, який використовує Хаар-подібні функції для виявлення транспортних засобів. Для оцінки відстані метод використовує кількість горизонтальних пікселів виявленої площі транспортного засобу. Регіони транспортних засобів, що претендують на вибір, вибираються з використанням характеристик Хаара, і лише регіони транспортних засобів виявляються в регіонах транспортних засобів-кандидатів відповідно до форми розподілу кромки через край Собеля. Відстань до виявленого транспортного засобу оцінювали за допомогою попередньо навченої бази даних відстаней, яка обчислює фактичну кількість пікселів на метр. Недоліком цього методу є низька точність, оскільки він не враховує зміни в роздільній здатності зображення та ступені спотворень камери

в реальних розрахунках. Крім того, розрахована база даних розрахункової відстані (БД) до 15 м, що обмежує її застосування в реальних дорожніх умовах.

Бертоцці та ін. [55] запропонував метод проектування двовимірного зображення на тривимірний реальний світ шляхом обчислення зворотного перетворення перспективи для оцінки відстані до транспортного засобу, виявленого на зображенні, отриманому з камери. Однак проектування двовимірного зображення на тривимірний простір є складним завданням для обробки в режимі реального часу, оскільки кількість обчислень збільшується на піксель.

З метою впровадження безпечного ADAS, Лі та ін. [65] і Інь та ін. [66] запропонував метод аналізу руху водія під час руху та зосередження уваги на водійському завданні. У запропонованому ними методі рівень ризику оцінювався з використанням інформації про рух рук водія. Хоча їх методи стосуються ненавмисної поведінки водія, пропонований метод пропонує метод, який зацікавлений у русі інших транспортних засобів.

У світі існують різноманітні методи визначення інтенсивності руху на автомобільних дорогах. Але в країнах, що розвиваються, таких як в Україні для визначення інтенсивності транспортного потоку все ще використовується візуальний облік [67]. Після визначення годинної інтенсивності руху, за допомогою поправкових коефіцієнтів визначають середньорічну добову інтенсивність руху. Це говорить про низький рівень розвитку інформаційних технологій у автотранспортній сфері в Україні.

Для того, щоб ефективно управляти ситуацією на дорозі, системам управління необхідно отримувати актуальні дані за невеликі проміжки часу. Такий показник, як середньорічна добова інтенсивність руху не дає можливості оцінити ситуацію на дорозі в конкретний момент часу, тому що розподіл кількості транспортних засобів на ділянці транспортного руху не є рівномірним протягом доби.

1.3 Інформаційні технології визначення інтенсивності транспортного руху

Проблема виявлення та опису заторів є життєво важливою для визначення шляхів обмеження її негативного впливу на функціонування транспортних систем. Сучасні рішення інтегрованих систем контролю дорожнього руху, таких як ITC, використовують широкий спектр пристроїв контролю дорожнього руху. Відеосистеми моніторингу набувають популярності та стають важливими джерелами інформації про дорожній рух. Технологія обробки зображень використовується для визначення даних про дорожній рух, таких як: присутність транспортного засобу на роз'їздах, кількість автомобілів та характеристики руху транспортних засобів [68].

Можна виділити декілька підходів до визначення заторів на основі відеоданих. У роботі [69] запропоновано метод аналізу руху для виявлення трьох типових станів руху - заторів, повільного та плавного. Стани класифікуються на основі значення MOFV (макрооптичної швидкості потоку), розрахованого за допомогою характеристичних векторів оптичного потоку, та значень щільності країв, оцінених за допомогою алгоритму Собеля для вилучення країв транспортного засобу. Порогові значення визначаються і використовуються для розрізнення стану дорожнього руху.

Автори роботи [70] використовують характеристики Хараліка для оцінки рівня перевантаженості транспорту. Енергетичні та ентропійні характеристики розраховуються за допомогою чотирьох матриць GLCM, що представляють горизонтальні, вертикальні та діагональні текстурні характеристики зображення. Складений показник, лінійна функція від обчислених значень, визначається для представлення щільності руху, безпосередньо пов'язаної із заторами. З метою зменшення обчислювального навантаження шкала сірого пікселів зменшується з 256 до 32 значень. Ефективність цього рішення обмежується виявленням заторів, тобто визначенням дуже великої щільності руху на спостережуваному місці події.

Цяо та Ши [71] пропонують використовувати функції зображення краю та текстури для вилучення параметрів руху, пов'язаних із заторами, тобто тривалості черги та зайнятості смуги на підходах до перехрестя. Шляхи руху визначаються як ROI (регіон, що цікавить), де застосовується виявлення краю Canny для отримання контурів транспортного засобу, а місцеві текстури двійкового візерунка обчислюються для опису транспортного засобу та дорожніх покриттів. Морфологічна обробка використовується для посилення присутності транспортних засобів та зменшення фонового шуму. Міжкадрова різниця береться для сегментації рухомих та нерухомих транспортних засобів. Результат аналізується для вилучення кінця черги та кількості транспортних засобів у черзі. Зайнятість на смузі визначається як відношення кількості пікселів транспортних засобів, які є як статичними, так і рухаються до числа пікселів ROI.

Сонг в роботі [72] пропонує середню інтенсивність віднімання фонового зображення для опису кількості транспортних засобів, що рухаються, і доводить лінійну залежність між цією характеристикою зображення та коефіцієнтом зайнятості. Стани дорожнього руху, пов’язані із заторами, класифікуються на основі характеристик, а також на основі характеристик освітленості, представлених гістограмою групи Гауса для фону. Середня швидкість руху транспортних засобів та класифікація місця події також включені до класифікатора. Класифікатор розрізняє стан дорожнього руху вдень, ввечері та вночі. Визначено три стани: незаблокований, повільний та заклинений.

Асмаа та ін. [73] використали глобальний оцінювач руху у відеосцені, щоб класифікувати дорожній рух за трьома категоріями: легкий, середній та важкий. Оцінювач визначається як довжина вектора руху блоку з мінімальною похибкою відповідності, використовується вікно пошуку 16 × 16 пікселів. Вектор разом із такими вилученими параметрами, як заповнюваність, середня швидкість транспортних засобів, потік руху, параметри щільності руху застосовуються до класифікаторів, K-найближчий сусід (KNN) та SVM

(Підтримує векторний апарат). Найкращі показники забезпечуються використанням SVM із макроскопічними параметрами середньої швидкості та щільності на основі векторів руху блоків.

Автори [74] використовують комбінацію особливостей текстури Хараліка та гістограми суми та різниці для вилучення міських районів з аерофотознімань, що корисно для оцінки діапазону областей руху в системах управління транспортом.

У роботі [75] представлений метод визначення заторів на основі особливостей текстури зображення, визначених Хараліком. Зображення витягуються з відеопотоку кожний 100-й кадр, це дає зображення змін дорожньої ситуації кожні 4 с. Спостерігається 40-метровий відрізок трисмугової дороги, і зображення відповідно одрізаються. Отримана послідовність спостереження перетворюється в тестові послідовності з різними відображеннями пікселів. Тестові послідовності: 256 зображень рівня сірого, 8 зображень рівня сірого та двійкові зображення, що містять контури об’єктів. Вони є основою для дослідження властивостей текстурних елементів та для проектування моделі перевантажень.

Підсумовуючи вище написане, у таблиці 1.1 представлено результати порівняльної характеристики існуючих методів визначення інтенсивності.

Таблиця. 1.1. Існуючі методи визначення інтенсивності

Візуальний облік	**Технічні засоби**	**Системи Weigh-in-Motion (WIM)**
Переваги		
Висока точність вимірювання	Надійна робота в умовах поганої оптичної видимості	Не потребують зупинки автомобілів
	Низька ціна пристроїв	Інформативність
Недоліки		
Низька економічна ефективність	Розпізнавання технічних засобів	Висока вартість
Довготривала присутність спостерігачів	Потреба в проведенні високовартісних робіт	Чутливість до електромагнітних перешкод

Результати представлені у таблиці 1.1 свідчать про значний технічний прогрес в реалізації технологій керування дорожніх рухом. Але на сьогоднішній день задача про визначення інтенсивності в Україні та в світі все ще залишається не розв’язаною через ряд недоліків існуючих засобів моніторингу транспортних систем – низька точність, складність реалізації системи та ін. Разом з тим існує величезна потреба у точному визначенні інтенсивності транспортного руху для підвищення якості управління транспортним рухом, що призведе до покращення економіки та екології міст і благополуччя населення.

1.4 Інформаційні технології прогнозування інтенсивності транспортного руху

Розвиток урбанізації спричиняє серйозні затори в багатьох столичних та великих містах світу. Таким чином, необхідно та неминуче проводити будівництво міської дорожньої інфраструктури та вдосконалене управління дорожнім рухом для задоволення попиту на подорожі [76]. Найефективніші стратегії зменшення заторів завжди залежать від точного та своєчасного прогнозування руху, наприклад, потоку руху для організації руху та оптимізації синхронізації сигналу, часу в дорозі або швидкості для наведення маршруту руху автомобіля.

З початку 1980-х років методика короткострокового прогнозування дорожнього руху стала однією з найважливіших складових ІТС, і вікно часу прогнозування коливається від декількох хвилин до кількох годин у майбутньому на основі геометрії дороги, інформації про дорожній рух , а також стратегії контролю тощо [77]. Оглядаючи літератури за останні кілька десятиліть, модель прогнозування можна приблизно класифікувати на дві категорії: одинарні моделі та комбіновані.

Перші, як правило, присвячені певним формулам, враховуючи поточну та минулу інформацію про дорожній рух. Існуючу літературу можна розділити на

дві підгрупи. Однією з категорій є параметричні моделі, які можна описати за допомогою кінцевої кількості параметрів, таких як модель експоненціального згладжування [78], алгоритм середнього історичного значення [79], авторегресивна інтегрована ковзна середня (ARIMA) [80], фільтр Калмана (KF) [81, 82] та модель теорії Грея (GM) [83]. Серед них GM підходить для прогнозування системи низької інформації та невизначеності, такої як транспортний потік [84]. Окутані та Стефанедес [85] почали застосовувати фільтр Калмана для прогнозування руху транспорту на дорожній мережі в Нагої, Японія. Більше того, Сан та ін. [86] розробив модель лінійної регресії для прогнозування потоку на автостраді US-290 у Х'юстоні, США, яка виявила, що вона перевершує метод k-найближчого сусіда та метод згладжування ядра. Інша категорія, непараметричні моделі, передбачає, що структура параметрів трафіку не є фіксованою і, в основному, дотримується статистичної регулярності в залежності від великих польових даних, таких як Support Vector Machine (SVM) [76], штучні нейронні мережі [87-89], непараметрична регресія [90], гауссова максимальна ймовірність [91]. Серед них штучна нейронна мережа (ANN) є одним із найбільш широко використовуваних методів, оскільки вона може фіксувати коливання трафіку [92]. Лян і Вей [93] змоделювали потік трафіку на автострадах на основі простих рекурентних мереж, а також саме мережі Elman. Тан та ін. [94] довів, що модель k-Найближчий сусід (k-NN) може перевершити ARIMA та модель експоненціального згладжування на основі польових даних Гуанчжоу, Китай. Пізніше Lv та ін. [95] вперше застосував Stacked Auto-Encoder (SAE) для короткострокового прогнозування потоку руху і навчив модель за алгоритмом Greedy Layer-Wise. Нещодавно Ю та ін. [96] проаналізували часово-просторові характеристики транспортного потоку, а потім застосували згорткову нейронну мережу (CNN) для прогнозування короткотермінового потоку на основі розділу розташування. Згодом Ксу та ін. [97] розробили штучний алгоритм рійної риби для оптимізації векторної регресії опори для прогнозування потоку. З розвитком комп’ютерних технік машинне навчання також використовувалось для прогнозування потоків на

основі величезних історичних даних. Наприклад, Даї та ін. [98] запропонували глибоке навчання для прогнозування трафіку, а саме DeepTrend 2.0, яке розглядає інформацію про багато датчиків як вхідну інформацію і одночасно формує прогнозовані результати для всіх датчиків. Тим часом Лі та ін. [99] запропонували підхід, що спирається на глибокі особливості, на наступних декількох етапах із використанням контрольованих методик навчання. Чжао та ін. [100] передбачили рух транспорту на чотирьох ділянках дороги в Пекіні за допомогою LSTM і виявили, що він перевершує ARIMA та RNN (періодична нейронна мережа). Полсон і Соколов [101] зазначають, що архітектури глибокого навчання здатні фіксувати нелінійні просторово-часові ефекти, що виникають внаслідок переходів між вільним потоком, пробоєм, відновленням і заторами в транспортному потоці.

На відміну від попередніх єдиних методів прогнозування, більшість комбінованих моделей можуть отримати набагато більше переваг, ніж однотипні окремі, завдяки використанню двох або більше переваг методів. Наприклад, деякі дослідники віддали перевагу вибору ANN як основної моделі інтеграції інших методів, таких як GM [102, 103], ARIMA [104], k-NN [105], підтримка векторної регресії (SVR) [106], алгоритм кластеризації [107] та простий статистичний підхід [108]. Крім того, Фенг та ін. [109] поєднали функцію вейвлету та екстремальну навчальну машину (ELM), щоб запропонувати короткостроковий прогноз, який перевершив ANN на основі польових даних канадського шосе. Нещодавно Ву та ін. [110] запропонував поєднане глибоке вивчення CNN-RNN, враховуючи тижневу / добову періодичність та просторово-часові характеристики транспортного потоку. Відповідно до однієї розкладеної періодичної послідовності та двочастинних випадкових для часових рядів потоку руху, Женг та ін. [111] запропонував гібридне прогнозування з моделями ANN, ε-SVR та LSTM на основі зворотного поширення (BP).

Незважаючи на те, що багато світових дослідників повідомляють про велику різноманітність методів прогнозування потоку руху, короткочасне

прогнозування потоку руху по міських сигнальних коридорах все ще залишається складним завданням, оскільки транспортні потоки мають багато невизначеності (наприклад, змінюються в часі, сильно коливаються, нелінійні та нестаціонарні). Зокрема, більшість досліджень присвячені одній моделі для цілодобового прогнозування шляхом кількісної оцінки зв'язку між предиктором та залежними вхідними змінними, що може призвести до переоснащення моделі через великі коливання потоку руху протягом годин.

1.5 Висновки

Висвітлено основні тенденції застосування інформаційних транспортних систем та розглянуто сучасні інформаційні транспортні системи виявлення транспортних засобів.

В даному розділі розглянуто існуючі транспортні системи визначення інтенсивності транспортного руху. Незважаючи на наявність розвинутих технологій керування дорожніх рухом, на сьогоднішній день задача про визначення інтенсивності в Україні та в світі все ще залишається не розв’язаною через ряд недоліків існуючих засобів моніторингу транспортних систем – низька точність, складність реалізації системи та ін. Разом з тим існує величезна потреба у точному визначенні інтенсивності транспортного руху для підвищення якості управління транспортним рухом, що призведе до покращення економіки та екології міст і благополуччя населення.

Розглянуто існуючі транспортні системи прогнозування інтенсивності транспортного руху. Хоча розроблено велику кількість методів, прогнозування транспортного потоку все ще лишається складним завданням і потребує розроблення нових точних методів.

З розвитком сучасних технологій комп’ютерного зору та зі збільшенням кількості відео-камер спостереження за дорожнім рухом з’являється все більше можливостей для побудови нових ITC для визначення інтенсивності

транспортних потоків, створення якої є актуальною задачею для підвищення якості управління транспортним рухом.

РОЗДІЛ 2
ПРОГРАМНИЙ КОМПОНЕНТ ВИЗНАЧЕННЯ ІНТЕНСИВНОСТІ ТРАНСПОРТНИХ ПОТОКІВ

Проведений порівняльний аналіз наявних ІТС призначених для визначення інтенсивності транспортних потоків засвідчив, що кожна з них має свої переваги та недоліки. З розвитком сучасних технологій комп'ютерного зору та зі збільшенням кількості відео-камер спостереження за дорожнім рухом з'являється все більше можливостей для побудови нової ІТС для визначення інтенсивності транспортних потоків, яка би задовольняла наступним вимогам: висока точність та надійність визначення інтенсивності транспортних потоків, низька вартість та зручність використання.

Для підвищення якості управління транспортним рухом є актуальним створення ІТС для визначення інтенсивності транспортних потоків з вище перерахованими перевагами, тому враховуючи це, основні завдання, які потрібно вирішити у дисертаційній роботі будуть наступними:

– розробити алгоритм обробки зображень з метою виявлення транспорту на наявній ділянці;

– розробити метод визначення показника завантаженості;

– розробити технологію визначення інтенсивності дорожнього руху за даними відеоряду;

– розробити метод визначення інтенсивності за послідовними значеннями показника завантаженості смуги дорожнього руху;

– розробити метод прогнозування показника завантаженості смуги дорожнього руху.

2.1 Метод визначення показника інтенсивності дорожнього руху на основі розробленої математичної моделі

Для оцінки інтенсивності дорожнього руху було розроблено метод визначення показника інтенсивності дорожнього руху на основі математичної моделі, що описує зв’язок між показником інтенсивності дорожнього руху та послідовними значеннями показника завантаженості смуги дорожнього руху TLCR.

Розроблений метод оснований на модулі Bioinspired бібліотеки OpenCV та нейронної мережі U-net для сегментації зображення. Алгоритм визначення показника інтенсивності дорожнього руху зображено на рис. 2.1 та складається з наступних етапів:

- отримання даних відеопотоку з відео-камер спостереження транспортного руху;
- створення dataset з зображеннями 1280x800 з відеоданих;
- обробка зображень в модулі Bioinspired для отримання ч/б зображень (1280x800) (детектор руху);
- обробка зображень для перетворення з 1280x800 до 256x256 для кожної смуги;
- навчання нейронної мережі U-net на зображеннях 256x256 отриманих даних записаних протягом декількох днів для розпізнавання рухомих об’єктів;
- за допомогою нейронної мережі U-net визначається показник завантаженості TLCR;
- визначення інтенсивності дорожнього руху за показником завантаженості TLCR.

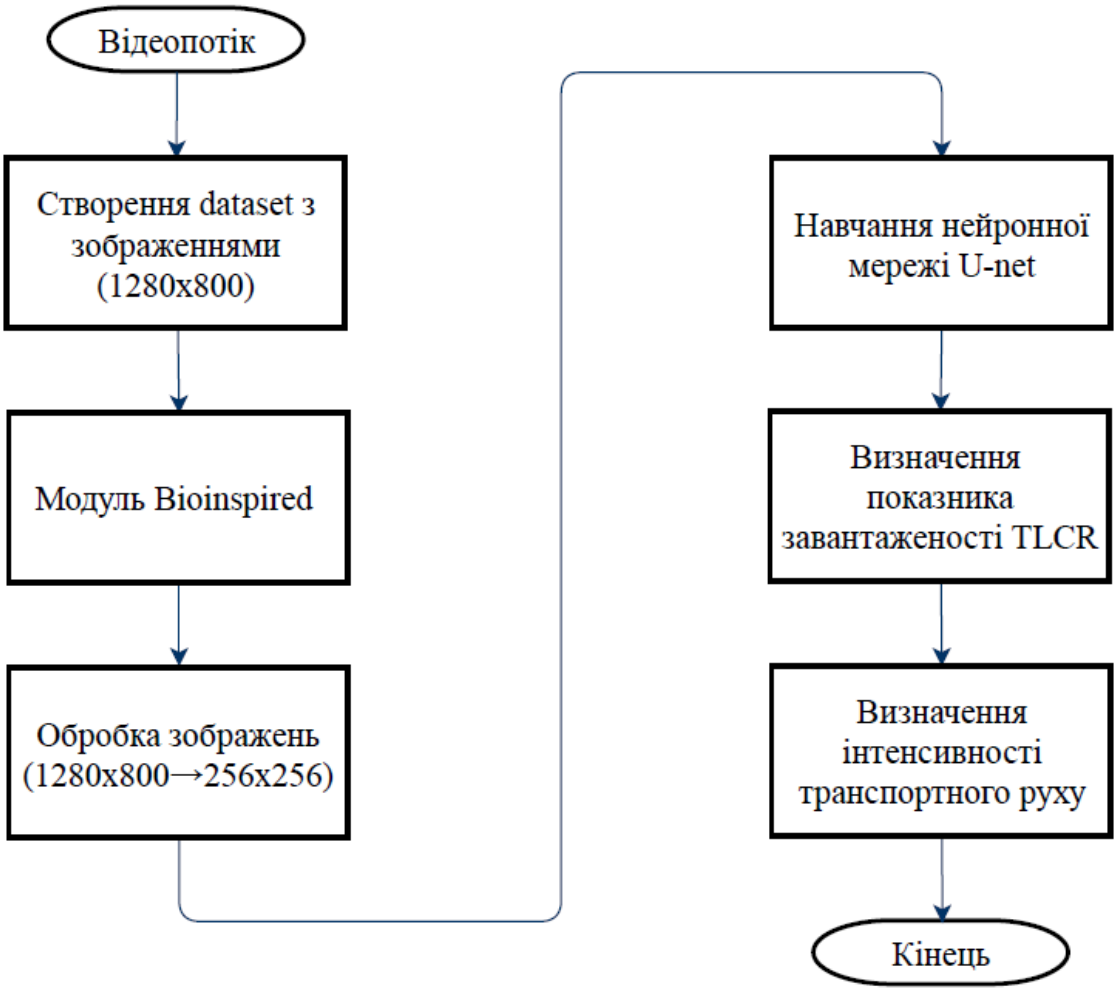


Рис. 2.1. Алгоритм визначення показника інтенсивності дорожнього руху

2.2 Коефіцієнт завантаженості смуги транспортного руху (TLCR)

В розділі 1 було описано недоліки основного показника визначення рівня заторів на дорогах параметра AADT (середньорічного щоденного трафіку). Головним недоліком AADT є неможливість адекватно визначати рівень заторів. Тому одним із головних задач даного дослідження було розробити метод визначення показника завантаженості смуги транспортного руху, який буде враховувати недоліки AADT.

Кожна відеокамера записує відео впродовж приблизно 1 хвилини і зберігає їх на сервері. Далі з відео отримуються окремі кадри для подальшої обробки. Кожен піксель кадру в кольоровому режимі відтінку сірого може мати

значення від 0 до 255, де 0 це чорний колір, а 255 білий. Координати розміщення області зображення, що відповідає смузі руху, задаються користувачем.

Для того, щоб отримати інформацію з відеопотоку про стан дорожнього руху, запропоновано показник, який отримав назву «коефіцієнт завантаженості смуги транспортного руху», або скорочено TLCR. Показник завантаженості характеризує зайнятість смуги дорожнього руху, або, іншими словами, коефіцієнт її використання транспортними засобами для переїзду. Коефіцієнт TLCR визначається як середнє значення значень пікселів, розміщених в області зображення, яка відповідає одній смузі, за формулою [117]:

$$C = \frac{\sum_{i \in A} V_i}{255 \cdot |A|} \tag{2.1}$$

де i – індекс пікселя області A,

$|A|$– кількість пікселів області A,

V_i – значення i-ого пікселя,

C – показник TLCR, який приймає значення в інтервалі [0,1].

Значення TLCR = 0 на виділеній області зображення досягається, коли транспортні засоби відсутні. При збільшенні транспортних засобів на виділеній області зображення значення показника TLCR буде збільшуватись. Значення TLCR, що близьке до одиниці, означає відсутність вільного місця на ділянці транспортного руху через зайнятість транспортними засобами і характеризує дорожній стан, близький до затору.

При визначенні показника завантаженості важливо правильно виділити область, що відповідає смузі руху. Автомобілі, що проїжджають по сусіднім смугам дорожнього руху, в залежності від геометричних властивостей та ракурсу камери можуть бути зараховані як автомобілі, що рухаються по досліджуваній області. Ракурс камери не завжди дозволяє використовувати прямокутник як область $\boldsymbol{A}$, що відповідає частині смуги руху з формули (2.1). Тому вирішено вдосконалити алгоритм, використовуючи будь-який чотирикутник замість прямокутника. Приклад зображено на рисунку 2.2.

Рис. 2.2. Область, яка виділена для визначення показника завантаженості смуги дорожнього руху

Для визначення кількості та суми значень пікселів у виділеній області смуги руху, використовується такий алгоритм:

- На першому кроці створюється зображення такого ж розміру, як і оригінальне, та розміщуємо на ньому фігуру, що відповідає досліджуваній області смуги дорожнього руху.
- На другому кроці наближено визначається прямокутник 1 навколо фігури за допомогою функції boundingRect бібліотеки opencv [112] та з використанням нейромережі вирізається з обробленого зображення другий прямокутник 2.
- На наступному кроці застосовується операція кон'юнкція для прямокутників 1 та 2, результатом якої є прямокутник 3 (рис. 2.3).
- Кількість пікселів в обраній області визначається на четвертому кроці кількістю ненульових пікселів в прямокутнику 1.
- На останньому кроці знаходиться сума значень пікселів в прямокутнику 3 за допомогою функції sumElems.

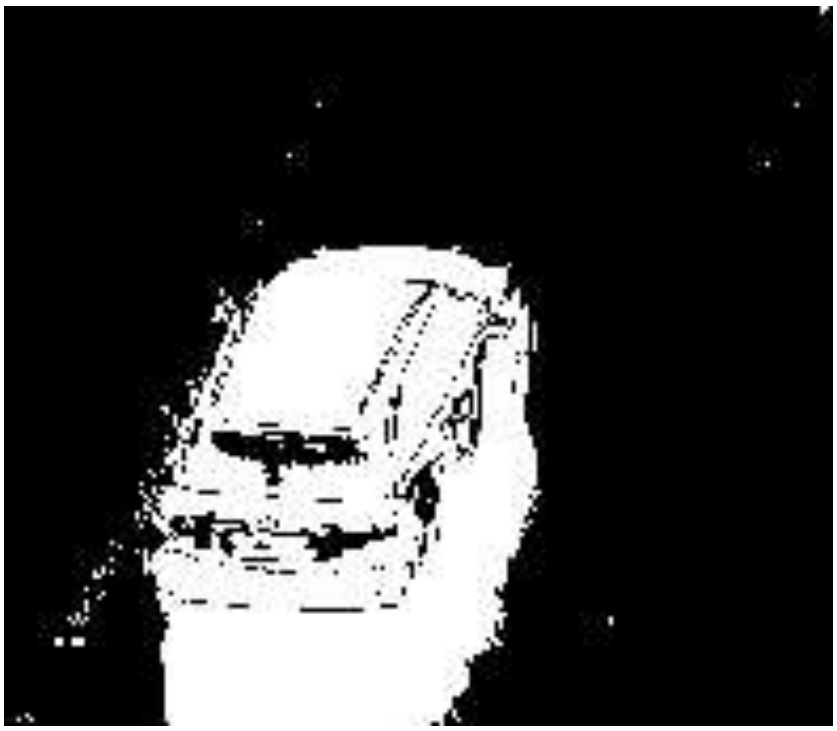


Рис. 2.3. Результат операції логічного «І»

В розділі 1 були описані недоліки відомих засобів та методів визначення інтенсивності дорожнього руху, тому в роботі запропоновано точну та надійну технологію визначення інтенсивності дорожнього руху за даними відеоряду.

Інтенсивність транспортного руху визначається за формулою:

$$I(A) = \frac{1}{T}\sum_{i=2}^{n}\begin{cases} 1, & if\, c_i > k and c_{i-1} \leq k otherwise \\ 0, & \end{cases} \quad (2.2)$$

де T – інтервал спостереження,

c – масив значень коефіцієнта завантаженості смуги руху (TLCR),

k – порогове значення, що визначається експериментально і є специфічним для відеокамери і смуги дорожнього руху.

Для визначення інтенсивності використовуються послідовні значення показника завантаженості TLCR. Транспортний засіб зараховується, якщо значення TLCR, що визначене з поточного кадру, більше ніж порогове значення та значення TLCR з попереднього кадру менше ніж порогове. Для правильного визначення інтенсивності важливо, щоб в досліджуваній області було не більше одного транспортного засобу. В іншому разі коливання значень TLCR при щільному потоці автомобілів не буде зафіксовано. Також важливо,

щоб досліджувана область не була занадто мала, тому що асиметричність частин транспортного засобу може привести до хибних коливань показника завантаженості.

Для визначення порогового значення використовується середньоквадратична похибка, розрахована на даних перевірочної послідовності, яка відома також як критерій регулярності [15]:

$$K = \frac{\sum_i \left(c_i^{real} - c_i^{model}\right)^2}{\sum_i \left(c_i^{real}\right)^2} \quad (2.3)$$

де c_i^{real} – фактичне значення інтенсивності,

c_i^{model} – обчислюване значення інтенсивності.

Визначення найкращого порогового значення складається з таких етапів:

1) Проведення візуального обліку та визначення фактичної інтенсивності транспортного руху за 2 години. Кожне значення – це інтенсивність за 1 хвилину.

2) Поділ даних на дві частини: навчальна послідовність даних та послідовність даних для перевірки.

3) Визначення найкращих значень $\boldsymbol{k}$ (порогове значення) за допомогою критерію, який визначається на навчальній послідовності даних в обраному інтервалі за формулою (3). При цьому обрано інтервал від 0,05 до 0,5 з кроком 0,005.

4) Визначення найкращого значення k за допомогою критерію, визначеного на перевірочній послідовності даних.

Для експериментальних даних отримано найкраще порогове значення $\boldsymbol{k}$=0,155, відповідне значення критерію на перевірочній послідовності даних дорівнює 0,000064, що свідчить про достатньо високу точність моделі. Показником якості моделі є критерій (3).

Для реалізації інформаційної технології визначення інтенсивності дорожнього руху використовувались наступні технічні пристрої: камера відеоспостереження та комп'ютер, з’єднані Інтернет-зв’язком.

Процес визначення інтенсивності транспортного руху за відеорядом складається з наступних етапів:

а) збір даних з камери відеоспостереження;

б) перетворення відео в послідовний набір зображень;

в) сегментація зображень для виділення транспортних засобів;

г) розподіл транспортних засобів по смугах дорожнього руху;

ґ) визначення показника завантаженості TLCR для окремої смуги руху на кожному зображенні;

д) визначення інтенсивності дорожнього руху з показників завантаженості;

е) визначення загального показника завантаженості;

є) запис показників в базу даних.

Програмне забезпечення складається з наступних модулів:

- Модуль збору даних.
- Модуль обробки зображень (сегментація).
- Модуль визначення показника завантаженості.
- Модуль визначення інтенсивності дорожнього руху.

При розробці програмного забезпечення використовувались наступні інструменти: мова програмування python, база даних sqlite, та бібліотеки tensorflow, opencv, numpy, pyside (Додаток А).

2.3 Програмна реалізація визначення руху об’єктів у відеопотоці

Розроблена програма написана на мові C++ з використанням бібліотеки OpenCV та компілятора MSVC 2017. Визначення руху об’єктів у відеопотоці реалізовано в двох варіантах:

- на основі порівняння двох кадрів;
- на основі модуля «bioinspired».

Кожен з цих варіантів отримує зображення з відеокамери: 30 кадрів за секунду. Приклад зображення представлений на рис. 3.2.

Рис. 3.2. Початковий кадр

2.3.1 Програмна реалізація визначення руху об’єктів у відеопотоці на основі порівняння двох кадрів

У найпростішому варіанті цього методу пошук відмінностей призводить до вивчення кожного пікселя на першому зображенні та перевірки, чи піксель видно на другому зображенні. Проблема цього підходу полягає в тому, що він лише позначає зміну, не вимірюючи її. Немає різниці, піксель став трохи темнішим або він має зовсім інший колір. Для вирішення цієї проблеми застосовано метод фільтрації «Розмивання Гауса».

Програмна реалізація складається з таких етапів:

1) Перетворення вхідного зображення з RGB в «Градацію сірого».

2) Застосування методу фільтрації «Розмивання Гауса» (рис. 3.3), щоб знизити рівень шуму.

Рис. 3.3. «Градація сірого» та фільтрація «Розмивання Гауса»

3) Отримання різниці між зображенням з пункту 2 та попереднім зображенням (рис. 3.4). Для цього потрібно поділити зображення на блоки та отримати середнє значення кольору для кожного блоку. Ділити на блоки потрібно з урахуванням оптимізації кількості обчислень. Наприклад, якщо зображення розміром 640x480 пікселів поділити на блоки розміром 10x10 і отримати з кожного блоку кольори 25 пікселів, то в результаті замість 300000 пікселів і ітерацій для аналізу отримаємо 3000 ітерацій і 75000 пікселів для аналізу [119]. Порівняємо дві таблиці кольорів (від попереднього та поточного кадру) і отримуємо різницю кольорів по кожному блоку в третій таблиці (рис. 3.5).

Рис. 3.4. Різниця між зображеннями

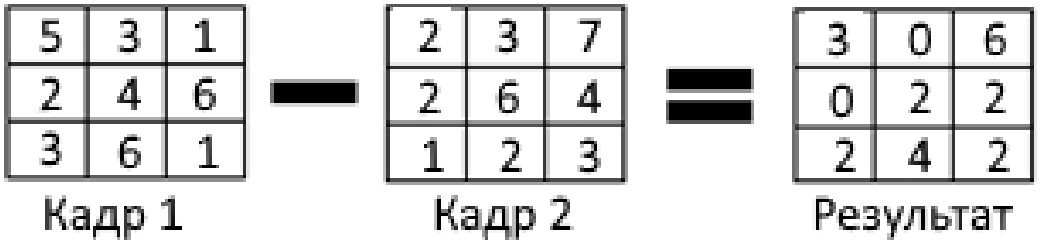


Рис. 3.5. Різниця між таблицями кольорів

4) Порівняння значення отриманої різниці з пункту 3 з деяким пороговим значенням. Експериментально було обрано число 25. Якщо значення більше ніж порогове то такий піксель належить рухомому об’єкту, інакше значення дорівнює 0 (рис. 3.6).

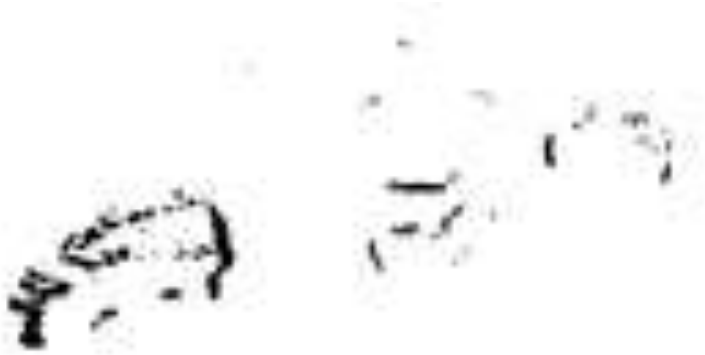


Рис. 3.6. Різниця між зображеннями після застосування порогового значення

2.3.2 Програмна реалізація визначення руху об’єктів у відеопотоці на основі модуля «bioinspired»

Модуль bioinspired [120] містить клас Retina, в якому знаходяться просторово-часовий фільтр двох інформаційних каналів (parvocellular pathway і magnocellular pathway) моделі сітківки ока. В основному нас цікавить magnocellular канал, який вже є детектором руху. Єдине, що нам потрібно, це отримати координати області зображення, де виявляється рух, і якось реагувати на перешкоди, які з’являються, коли детектор руху показує статичне зображення.

Реалізація складається з таких етапів:

1) Перетворення вхідного зображення з RGB в «Градацію сірого».

2) Запуск модулю «magnocellular» (рис. 3.7).

3) Перевірка медіанного фільтру від ентропії, для того щоб при відсутності будь якого руху не реагувати на шум. В дослідженні було використано значення 0,60.

4) Порівняння результату з пороговим значенням 90 (рис. 3.8).

Рис. 3.7. Результат модулю «magnocellular»

Рис. 3.8. Результат після застосування порогового значення

Алгоритм застосовується для кожного зображення отриманого в результаті методу детекції руху. В даному алгоритмі кожен піксель перевіряється на належність відрізку MN, який відповідає одній смузі руху, та складається з точок M(x1;y1) та N(x2;y2).

Для перевірки належності точки відрізку MN використовується формула:

$$\frac{a-x_1}{x_2-x_1} = \frac{b-y_1}{y_2-y_1} \tag{3.1}$$

де a, b – координати пікселя, x_1, y_1 – координати точки M, x_2, y_2 – координати точки N.

Якщо точка (a, b) задовольняє рівності (2.1), то автомобіль перетнув уявну смугу. Зі значення лічильника *count* машин, що перетнули смугу, збільшується на одиницю тільки за умови, що попередня машина вже перетнула смугу (див. рис. 3.9). Одразу після того, як фіксується перетин автомобілем уявної смуги, програма переходить в режим очікування і не фіксує нових автомобілів для того, щоб не фіксувати перетин смуги новим автомобілем, доки попередній автомобіль повністю не перетне смугу.

Порівняння точності методів визначення кількості транспортних засобів знаходиться в розділі 4.

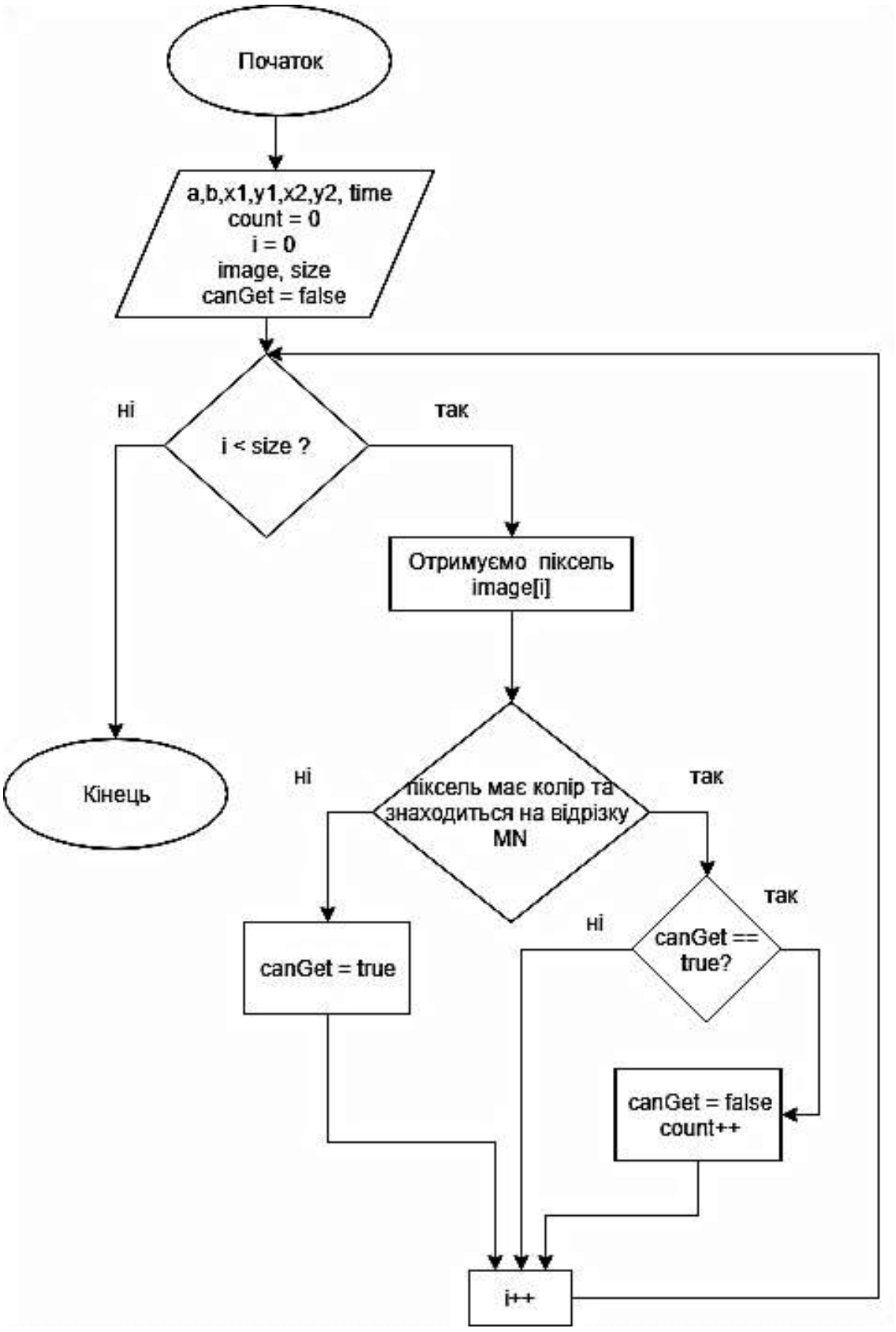


Рис. 3.9. Блок схема алгоритму визначення кількості машин

2.4 Висновки

В роботі вперше було розроблено метод визначення показника завантаженості смуги транспортного руху, який отримав назву TLCR (Traffic Lane Congestion Ratio), для оцінки заторів на одній смузі. Розроблено алгоритм обробки зображень з метою виявлення транспортних засобів на наявній ділянці. Запропоновано метод визначення інтенсивності дорожнього руху за послідовними значеннями показника завантаженості смуги дорожнього руху за даними відеоряду.

Правильно визначені значення параметрів завантаженості смуги транспортного руху TLCR та інтенсивності дорожнього руху дозволить системі управління знайти оптимізовані параметри та уникнути заторів.

Розроблено програмний компонент для визначення інтенсивності дорожнього руху в двох реалізаціях. Експериментально доведено значну перевагу реалізації на основі модуля bioinspired над реалізацією на основі порівняння двох кадрів. Програмний компонент визначення інтенсивності транспортного руху, який розроблений, є складовою частиною інформаційної системи управління транспортним рухом.

РОЗДІЛ 3
ТЕХНОЛОГІЯ ВИЗНАЧЕННЯ ІНТЕНСИВНОСТІ ДОРОЖНЬОГО РУХУ ЗА ДАНИМИ ВІДЕОРЯДУ

Для дослідження отримувались дані з веб-сайту videoprobki.ua. Для обробки зображень реалізовано нейромережу U-Net. Нейромережа вирішує задачу сегментації, що дозволяє виділити пікселі, які мають спільні візуальні характеристики [117]. Зазвичай такі нейромережі використовують для обробки біомедичних зображень, оскільки для навчання не потрібна велика кількість даних [118]. Результат обробки зображення смуги дорожнього руху нейромережею показано на рисунку 3.1.

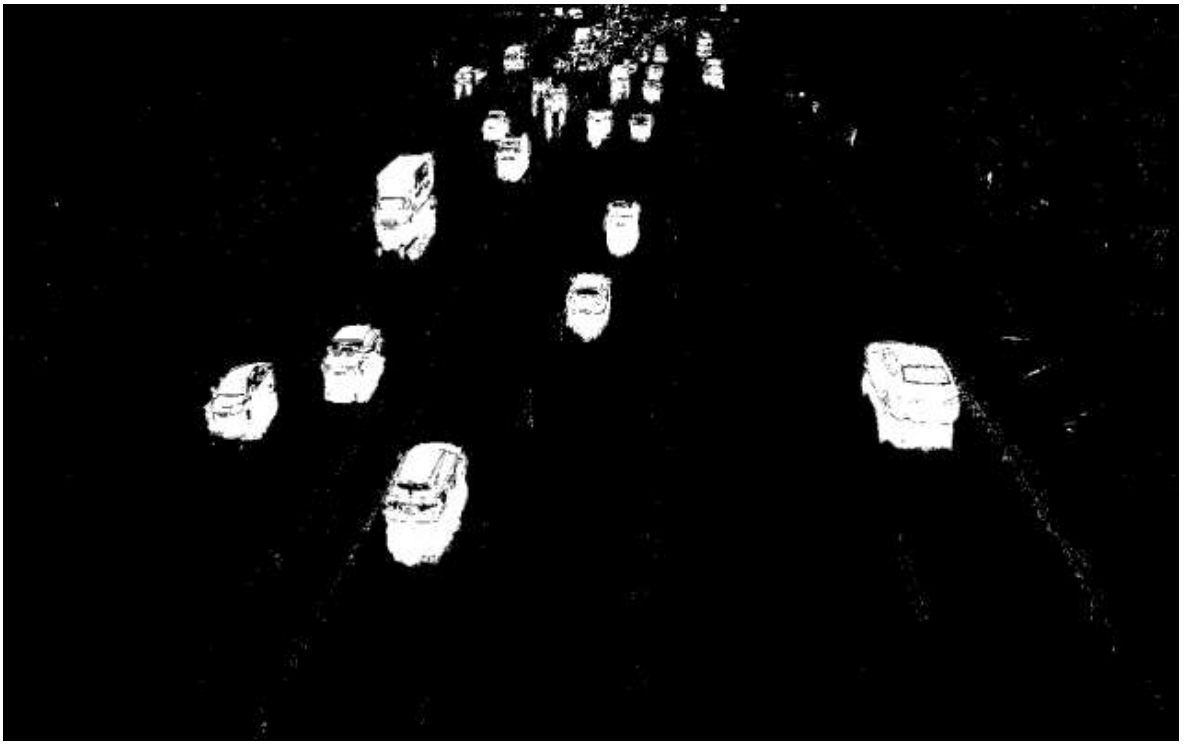


Рис. 3.1. Результат обробки зображення смуги дорожнього руху мережі U-NET.

3.1 Розпізнавання образів за допомогою бібліотеки OpenCV

OpenCV є широко відомою бібліотекою [112] програмних засобів для обробки та аналізу вмісту зображень, у тому числі розпізнавання об’єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об’єктів, перетворення зображень з використанням методів машинного навчання, виявлення загальних елементів на різних зображеннях. Перевагою використання OpenCV є широкий вибір цільової платформи, мови реалізації і відсутність обмежень на поширення готових продуктів з її використанням. Бібліотека OpenCV розділена на невеликі модулі по їх функціональному використанню:

- opencv_core містить базові структури та обчислення (математичні функції, генерація псевдовипадкових чисел, DFT, DCT, введення / виведення в XML і т.п.);
- opencv_imgproc містить методи обробки зображень (фільтри, перетворення і т. д.);
- opencv_highgui реалізує простий GUI, завантаження / збереження зображень та відео;
- opencv_ml містить методи та моделі машинного навчання (SVM, дерева прийняття рішень і т. д.);
- opencv_features2d реалізує дескриптори (SURF);
- opencv_video призначений для аналізу руху і відстеження об’єктів (оптичний потік, шаблони руху, усунення фону);
- opencv_objdetect призначений для детекції об’єктів на зображенні (вейвлети Хаара, HOG і т. д.);
- opencv_calib3d містить методи для калібрування камери, пошук стерео-відповідності і елементи обробки тривимірних даних;
- opencv_flann є бібліотекою швидкого пошуку найближчих сусідів (FLANN);
- opencv_contrib є супутнім кодом, ще не готовим для застосування;

- opencv_legacy є застарілим кодом, який збережено заради зворотної сумісності;
- opencv_gpu реалізує прискорення деяких функцій OpenCV за рахунок використання CUDA (NVidia).

Окрім основних модулів, є додаткові, які потрібно завантажувати окремо. Одним з таких модулів є «bioinspired». Цей модуль забезпечує елементи керування моделлю Gipsa/Listicm сітківки ока на основі невід'ємного просторово-часового фільтру, що моделює два основні інформаційні канали сітківки:

- Parvo - фовеальне бачення для детального кольорового зору;
- Magno - периферичне бачення для виявлення чутливих перехідних сигналів (рух та події).

Модель походить від роботи Дженні Еро, яка включає полярні трансформації Бартельмі Дюрете [113, 114]. В моделі високочастотний просторовий та часовий шум фільтрується. Обидва вихідні шляхи Parvo та Magno використовують результат фільтрації. Зменшення шуму забезпечує невід'ємну просторово-часову фільтрацію. На виході Parvo посилюються статичні текстури, а шум фільтрується (на відео, часовий шум добре видаляється). Однак, рухливі текстури згладжуються [115]. Тоді, деталі рухомих об'єктів можна посилити, лише якщо сітківка стежить за ними і зберігає їх статичними. На виході Magno це дозволяє більш точно виявляти події (рух, зміни) з послабленими шумовими помилками навіть у складних умовах освітлення. В якості компромісу вихід Magno є низьким сигналом просторової частоти і дозволяє точно визначати спрацьовування подій.

Модель може бути використана як етап попередньої обробки з метою виконання аналізу текстури з покращеним співвідношенням сигналу та шуму і покращеними деталями. Для полегшення використання в програмах два канали моделі сітківки (Magno, Parvo) застосовуються на всіх вхідних зображеннях. Це не відповідає дійсній топології сітківки, але це практично з точки зору обробки зображень.

Для розпізнавання образів можуть бути використані наступні методи:

1) Метод фільтрації. Зображення, що отримані на виході оптико-електронних перетворювачів, містять сторонню інформацію (перешкоди). При аналізі об’єктів на складному зображенні, фон також є перешкодою. Фільтрація допомагає виділити області, які цікаві для розпізнавання, без їх аналізу. Результатом фільтрації є оцінка корисного сигналу зображення. Зображення являє собою двовимірну функцію просторових координат, що змінюється повільніше, ніж двовимірна функція, яка описує перешкоду. Тому при оцінці корисного сигналу в кожній точці кадру розглядають околицю цієї точки (деяку множину сусідніх з нею точок), використовуючи загальні характеристики сигналу в цій околиці. В інших випадках ознакою корисного сигналу є різкі перепади яскравості.

2) Бінаризація. Операція порогового поділу, яка в результаті отримує бінарне зображення, має на меті радикальне зменшення кількості інформації, що міститься в зображенні. В процесі бінаризації вихідне півтонове зображення, що має певну кількість рівнів яскравості, перетворюється в чорно-біле зображення, пікселі якого мають тільки два значення 0 і 1.

3) Виділення контурів. Перетворення необхідне в тому випадку, коли є досить складне зображення і, використовуючи малі інструменти бібліотеки комп’ютерного зору, необхідно виділити об’єкти на цьому зображенні та потім з ними працювати.

4) Міжкадрова різниця. Використовується поблочне порівняння кадрів, при якому зображення розбивається на окремі блоки. Порівняння між кадрами проводиться на рівні блоків відповідно до обраного критерієм. Перший кадр (базовий) стискається незалежно від інших. Наступні кадри стискаються тільки в обсязі блоків, що змінилися. Блоки, які вважаються незмінними запозичуються з попереднього (базового) кадру [116].

3.2 Використання нейромережі U-net для обробки відеоряду дорожнього руху

Нейронна мережа U-net була вибрана через високу ефективність та результативність в медичній сегментації зображень та має наступні переваги: а) надійна у невеликих наборах даних; б) підтримує хороший баланс між ефективністю та результативністю сегментації, що забезпечує ефективну роботу загальної системи; в) добре працює як у сегментації, так і в локалізації. В роботі [16] U-net продемонстрував кращі результати при сегментації об'єктів на зображенні в порівнянні з іншими трьома методами глибокого навчання для виявлення об'єктів на зображенні: метод Locality Sensitive (LSM), повністю згорнута мережа (FCNN) та YOLO. Ці переваги роблять мережу U-net потужним засобом для вирішення завдань сегментації зображень.

Вибрана нейронна мережа U-net для обробки відеоряду є структурованою згортковою нейронною мережею, і як випливає з назви за своєю структурою подібна до типу U (структура мережі зображена на рис. 3.10). Зазвичай U-net мережа використовує методи кодування та декодування для злиття базової інформації та інформації високого рівня. У статті [118] представлений підхід до сегментації медичних зображень. Однак представлений підхід може бути використаний для інших завдань сегментації, включаючи зображення транспортного руху.

Для навчання нейронної мережі використовувались два алгоритми оптимізації: Адам та SGD. Адам - це алгоритм оптимізації, який можна використовувати замість класичної процедури стохастичного градієнтного спуску для ітеративного поновлення ваг мережі на основі навчальних даних. Алгоритм Адам масштабує значення градієнта. Щоб знати, як часто змінюється градієнт, автори алгоритму Дідерік Кінгма і Джимі Бааз пропонують використовувати середню нецентровану дисперсію. Головна перевага алгоритму Адама полягає в тому, що він дозволяє достатньо швидко досягти

достатньо точних результатів. Алгоритм Адама перетворює градієнт наступним чином [121]:

$$S_t = \alpha \cdot S_{t-1} + (1-\alpha) \cdot \nabla E_t^2, S_0 = 0,$$

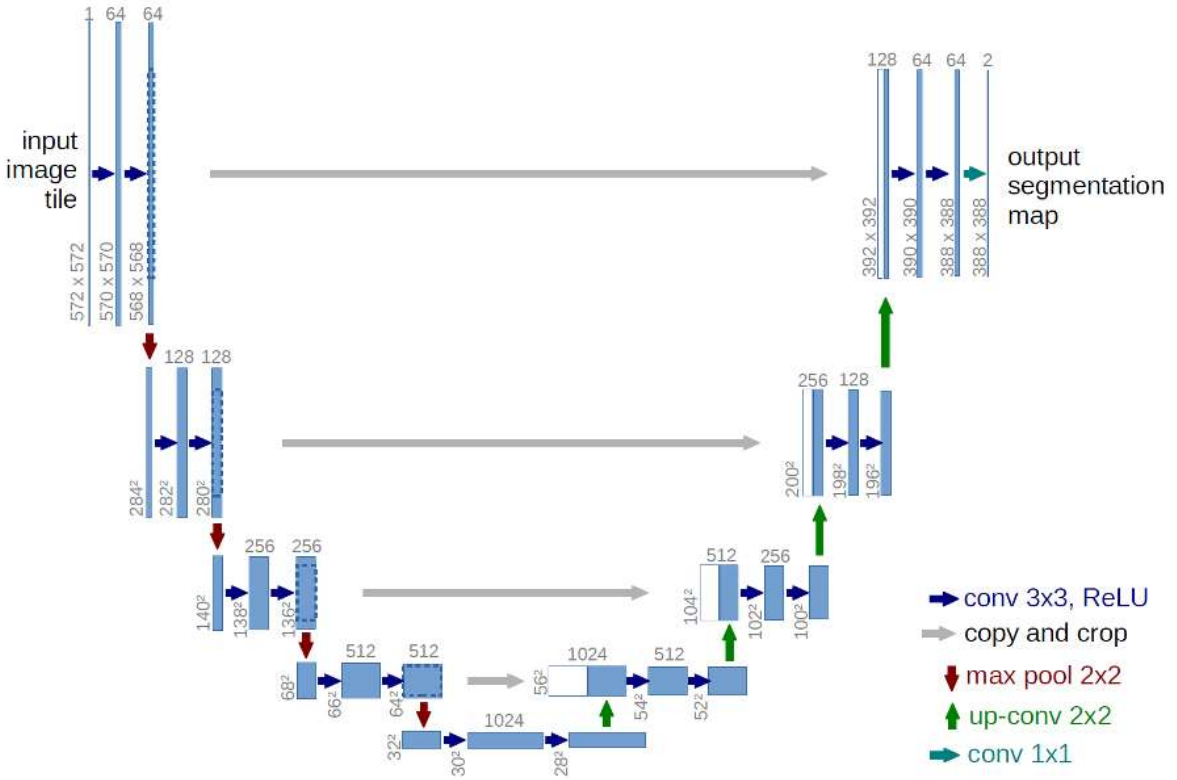


Рис. 3.10. Схема U-net моделі [118]

$$D_t = \beta \cdot S_{t-1} + (1-\beta)\nabla W_{t-1}^2, D_0 = 0,$$

$$g_t = \frac{\sqrt{D_t}}{\sqrt{S_t}} \cdot \nabla E_t, \qquad (3.2)$$

$$\nabla W_t = \eta \cdot (g_t + \rho \cdot W_{t-1}) + \mu \cdot \nabla W_{t-1},$$

де η - швидкість навчання,

∇Et - градієнт функції втрат,

μ - коефіцієнт моменту,

ΔW_{t-1} - зміна ваги у попередній ітерації,

ρ - коефіцієнт регуляризації,

W_{t-1} - ваги на попередній ітерації,

α = 0,999, β = 0,9,

S_t - оцінки першого моменту (середнє значення),

D_t - оцінки другого моменту,

g_t - градієнт.

Тренінг з набору даних, що складається з 10000 зображень, проводився з наступними параметрами: кількість епох встановлено на 5, розділення перевірки встановлено на 0,1, значення метрики встановлено на «перетин над об’єднанням».

Для навчання мережі було вирішено використовувати метод SGD (стохастичний градієнтний спуск). SGD є одним із методів оптимізації, а саме оптимізатором першого порядку, що означає, що він базується на аналізі градієнта об'єкта. Тому, з точки зору нейронних мереж, він часто використовується разом із зворотним розповсюдженням для ефективного оновлення. Для розрахунку SGD потрібно розрахувати градієнт моделі, і тут зворотне розповсюдження є ефективним методом для розрахунку градієнта. На діаграмі нейронної мережі ви можете побачити кількість шарів. З метою вдосконалення мережі було вирішено розрізати вхідні зображення на сегменти розміром 128 на 128 пікселів (рис. 3.11). На рис. 3.12 зображена схемафільтрації, яка відбувається на кожному шарі нейронної мережі [122].

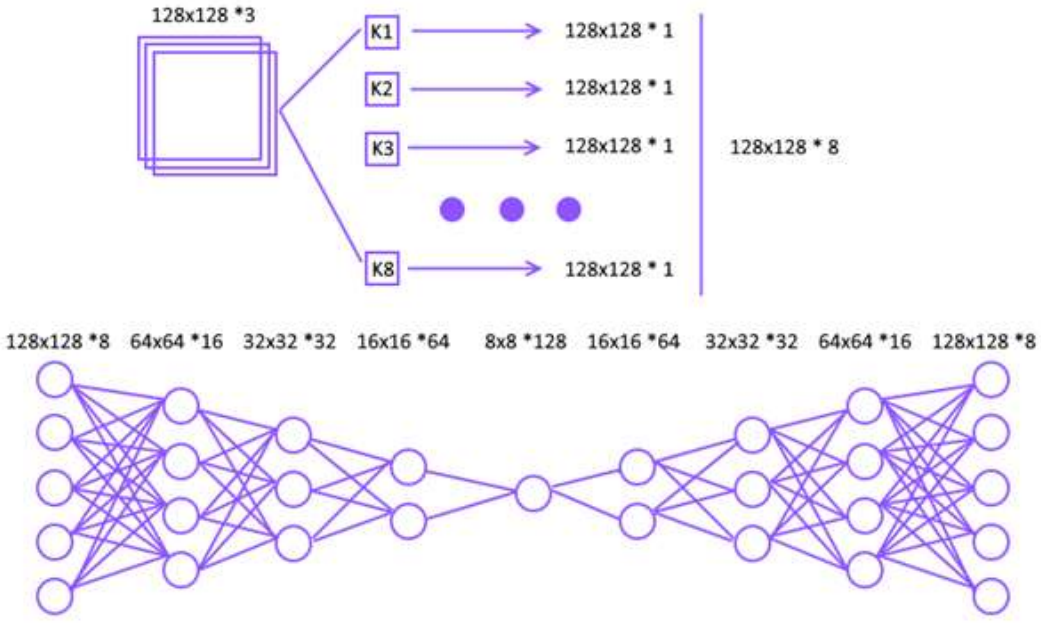


Рис. 3.11. Структура нейронної мережі [122]

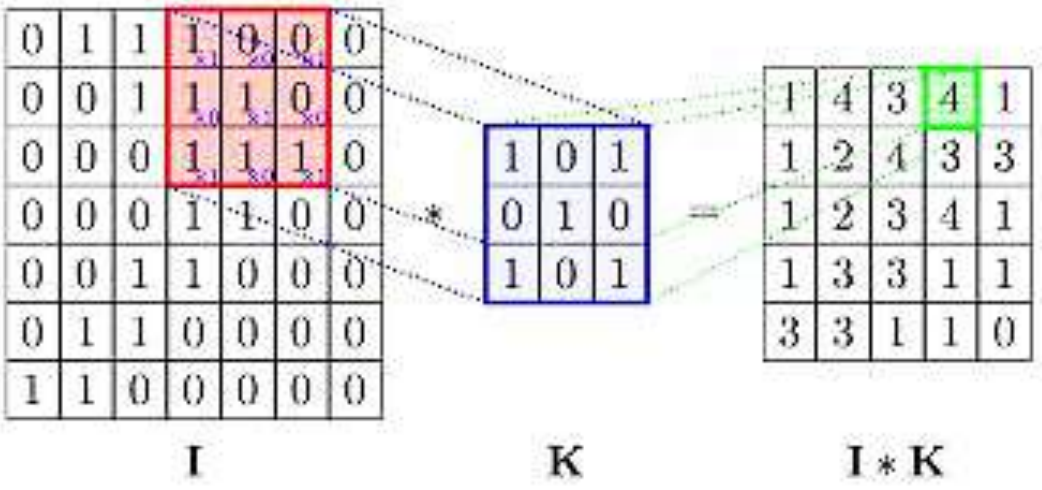


Рис. 3.12. Схема фільтрації [122]

3.3 Формування набору даних для навчання рекурентної нейронної мережі для прогнозування TLCR

Для вирішення завдання отримувати прогнозовані дані про інтенсивність дорожнього руху було обрано рекурентну нейронну мережу (RNN). Рекурентні нейронні мережі можуть навчатись використовувати минулу інформацію для прийняття рішень, але звичайні прості моделі RNN мають обмеження у роботі з довготривалими послідовностями. Тому була розроблена покращена версією базового блоку RNN – LSTM для якіснішого захоплення довготривалих послідовностей.

Повна мережа LSTM складається з декількох шарів блоків LSTM. Блок LSTM є базовою одиницею моделі LSTM з її здатністю вивчати довгострокові залежності. Кожна комірка LSTM складається із прихованої області $\boldsymbol{a_t}$ та області пам’яті $\boldsymbol{c_t}$, яке оновлюється після кожної ітерації. На відміну від базових RNN, при відображенні вхідної послідовності $\boldsymbol{x}$ у відповідну вихідну послідовність $\hat{\boldsymbol{y}}$ комірка LSTM складається в основному з чотирьох шлюзів, а саме: шлюз забування $\boldsymbol{f_t}$, шлюз входу $\boldsymbol{i_t}$, шлюз кандидата $\hat{\boldsymbol{c}}_{\boldsymbol{t}}$ та шлюз виходу $\boldsymbol{o_t}$, як показано в структурі, представленої на рис. 3.1. Ці шлюзи регулюють інформаційний потік, що допомагає фіксувати великі залежності.

Кожен зі шлюзів в комірці LSTM має наступне призначення:

1. *Шлюз забування* $\boldsymbol{f_t}$: Цей шлюз вирішує, яка інформація про попередній стан пам'яті $\boldsymbol{c_{t-1}}$ повинна бути переадресована далі або видалена з обчислення мережі. Даний шлюз розглядає $\boldsymbol{a_{t-1}}$ та $\boldsymbol{x_t}$, і виводить число в області комірки $\boldsymbol{a_{t-1}}$ від 0 до 1 для кожного значення. Шлюз забування визначає яку інформацію про показник завантаженості транспортного руху TLCR слід зберігати в стані пам'яті, і визначає коли видаляти інформативні часові залежності з пам'яті.

2. *Шлюз входу* $\boldsymbol{i_t}$: Цей сигмоподібний шлюз обчислює одну частину інформації, яку необхідно зберігати в області комірки.

3. *Шлюз кандидата* $\boldsymbol{\hat{c}_t}$: Цей шлюз використовує шар ***tanh*** для створення нового набору нелінійних функцій. Значення, які були отримані зі шлюзів входу та кандидата, об'єднуються для оновлення прихованої області комірки $\boldsymbol{c_t}$. Ці обчислені значення покращують лінійні та нелінійні здібності до навчання комірки LSTM та приймають рішення, яка інформація про особливості транспортного руху буде зберігатись в області комірки $\boldsymbol{c_t}$.

4. *Шлюз виходу* $\boldsymbol{o_t}$: Цей шлюз приймає рішення, яка інформація про показник завантаженості транспортного руху TLCR буде надсилатись на вихід $\boldsymbol{a_t}$. Цей результат буде ґрунтуватись на значенні області комірки $\boldsymbol{c_t}$, але буде відфільтрованим варіантом. Спочатку в сигмоподібному шарі приймається рішення, які частини $\boldsymbol{a_{t-1}}$ прихованої області будуть переадресовані до виходу. І в кінці, область комірки, яка проштовхується через активацію ***tanh***, множиться на вихід сигмоподібного шлюзу, який позначає вихід комірки $\boldsymbol{a_t}$. Цей рівень відповідає за обчислення вихідної області на основі попередньої області.

Рівняння (1), (2), (3), (4), (5) та (6) ілюструють, як комірка LSTM може зберігати пам'ять протягом тривалого часу та при необхідності пропускати інформацію. За допомогою рівнянь (1), (2) та (3) здійснюється керування шлюзом входу, шлюзом забування та шлюзом виходу відповідно.

$$\boldsymbol{u_t = \sigma\,(W_{ua}\,a_{t-1} + W_{ux}\,x_t + b_u)} \quad (1)$$

$$\boldsymbol{f_t = \sigma\,(W_{fa}\,a_{t-1} + W_{fx}\,x_t + b_f)} \quad (2)$$

$$\boldsymbol{o_t = \sigma\,(W_{oa}\,a_{t-1} + W_{ox}\,x_t + b_o)} \quad (3)$$

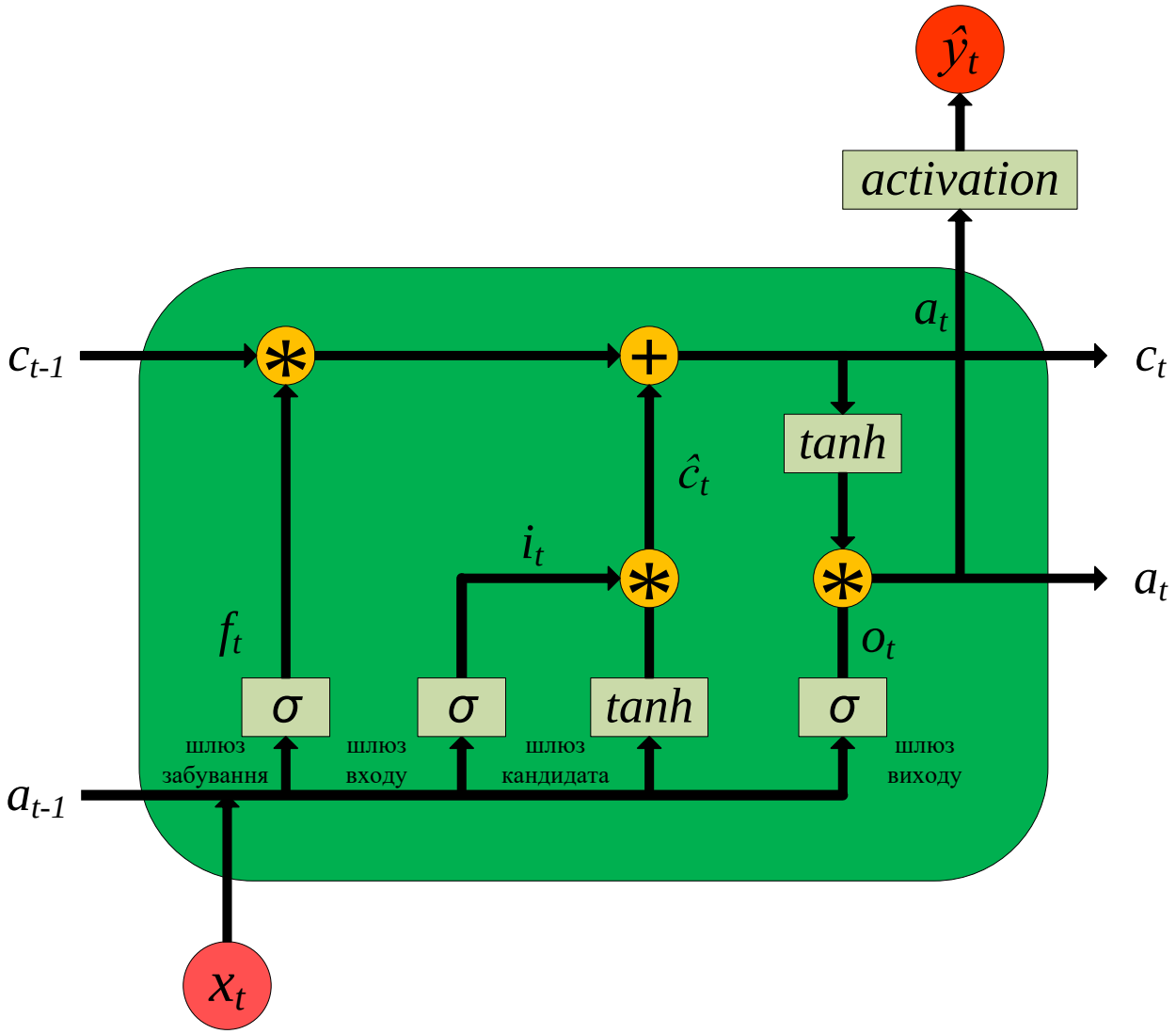


Рис. 3.1. Внутрішня структура блоку LSTM

За допомогою рівнянь (4), (5) та (6) здійснюється оновлення області комірок пам'яті c_t та виходу a_t.

$$\hat{c}_t = \tanh\left(W_{ca}\, a_{t-1} + W_{cx}\, x_t + b_c\right) \quad (4)$$

$$c_t = u_t \times \hat{c}_t + f_t \times c_{t-1} \quad (5)$$

$$a_t = o_t \times \tanh\left(c_t\right) \quad (6)$$

де W_{ua}, W_{ux}, W_{fa}, W_{fx}, W_{oa}, W_{ox}, W_{ca}, W_{cx}, b_u, b_f, b_o та b_c – всі параметри, що піддаються навчанню, де W_{ua}, W_{ux}, W_{fa}, W_{fx}, W_{oa}, W_{ox}, W_{ca} та W_{cx} – це зважені матриці, що регулюють зв'язок від відповідних входів до прихованого шару, тоді як b_u, b_f, b_o та b_c є зміщеннями. Параметр σ відноситься до сигмоподібної функції, а $\tanh$ – до гіперболічної дотичної функції, і обидві вони є нелінійними функціями активації, що визначаються наступними формулами:

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (7)$$

$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad (8)$$

У процесі керування шлюзом забування, як показано в рівнянні (2), спочатку він отримує інформацію як з поточного кроку, що вводиться $\boldsymbol{x_t}$, так і з результату попереднього кроку $\boldsymbol{a_{t-1}}$. Потім об'єднана інформація передається у функцію сигмоїдної активації, щоб вирішити, скільки інформації буде видалено із області комірки. Сигмоїдна функція виводить число від 0 до 1 для кожного числа в області комірки $\boldsymbol{c_{t-1}}$. Крім того, шлюз входу спрямований на вирішення, яка нова інформація буде зберігатися до стану комірки, де $\boldsymbol{u_t}$ діє як фільтр, а $\boldsymbol{\hat{c}_t}$ є кандидатом на заміну комірки пам'яті $\boldsymbol{c_{t-1}}$. Попередній стан комірки $\boldsymbol{c_{t-1}}$ спочатку множиться на $\boldsymbol{f_t}$, а потім додається до відфільтрованого кандидата $\boldsymbol{\hat{c}_t}\times u_t$, щоб отримати новий стан комірки $\boldsymbol{c_t}$. Нарешті, елементне множення фільтра $\boldsymbol{o_t}$ та оновленої клітинної пам'яті $\boldsymbol{c_t}$ генерує останні результати $\boldsymbol{a_t}$. Це демонструє, як функціонує одна комірка LSTM і чому вона може зберігати довгострокові спогади.

Для того, щоб навчити нейронну мережу LSTM для прогнозування TLCR, було зібрано дані записів дорожнього руху протягом одного тижня.

Набір даних для навчання мережі для прогнозування на *n* хвилин складається з:

- значення TLCR тиждень тому + *n* хвилин,
- значення TLCR добу назад + *n* хвилин,
- значення TLCR - *n* хвилин,
- поточне значення TLCR,
- значення TLCR + *n* хвилин

$$x^{t+1} = F\left(x^{t-60\cdot24\cdot7+1},\ x^{t-60\cdot24+1}, x^{t-1},\ x^{t}\right).$$

Для кожного значення TLCR в списку вище виконується метод змінного середнього:

$$x^{t} = \frac{1}{n}\sum_{i=t-n}^{t} V(i)$$

де t - час у хвилинах,

$V(i)$ - Значення TLCR в i-у хвилину,

n - розмір вікна, період згладжування.

Так як більшість існуючих систем управління транспортним рухом використовують інтенсивність як основну характеристику, необхідно прогнозувати саме цей параметр. З прогнозованого середнього значення завантаженості можливо отримати середнє значення інтенсивності лише в тому випадку якщо прогнозовані значення показника завантаження будуть отримуватись хоча б за кожну секунду. Через те що є залежність між інтенсивністю та показником TLCR, то можна використати рекурентну нейромережу LSTM і для прогнозування інтенсивності.

3.4 Висновки

Експериментально доведено перевагу використання нейронної мережі U-net для задачі визначення інтенсивності руху та показника завантаженості TLCR.

Розроблена технологія визначення інтенсивності дорожнього руху за даними відеоряду, що надходять з відеокамери спостереження. Було вдосконалено алгоритм визначення показника завантаженості транспортної ділянки TLCR надає можливість враховувати тільки автомобілі, які рухаються по досліджуваній смузі. Розроблений метод визначення інтенсивності дорожнього руху на основі послідовних значень показника завантаженості має наступні переваги над іншими подібними системами: швидкість обробки даних, точність, відсутність необхідності додаткового обладнання (наприклад датчики) та низька вартість.

Розроблено алгоритм та інформаційну систему для довгострокового прогнозування показника завантаженості транспортної ділянки TLCR для подальшої оцінки стану дорожнього руху. Інформаційна система прогнозування базується на моделі навчання з рекурентною нейронною

мережею LSTM. Розроблена система навчається на тренувальному відео отриманих з камер дорожнього руху записного протягом одного тижня для прогнозування показника завантаженості транспортної ділянки TLCR для кожного дня тижня.

РОЗДІЛ 4
ТЕХНОЛОГІЯ ПРОГНОЗУВАННЯ ІНТЕНСИВНОСТІ ДОРОЖНЬОГО РУХУ

4.1 Визначення інтенсивності транспортного потоку

В дослідженні точності двох реалізацій методів детекції руху: «Порівняння кадрів» та модуль bioinspired, які були описані в розділі 3.1використанодані з камери дорожнього руху, що знаходиться в м. Києві на перехресті вулиць «Олени Теліги» та «Дорогожицька». На рис. 3.6 та 3.8 представлені результати застосування методу детекції руху. Для того, щоб визначити кількість машин, які проїхали по одній смузі руху розроблено алгоритм, блок-схема якого представлена вище (див. рис. 3.9).

В таблиці 4.1 показано порівняння результатів застосування методу визначення інтенсивності транспортних потоків у двох реалізаціях. Відносна похибка визначається за формулою:

$$\varepsilon = \frac{\Delta\alpha}{\alpha} \cdot 100\%.$$

Таблиця. 4.1. Порівняння точності визначення інтенсивності транспортного руху

	Час, год.	Фактична кількість машин	Визначена кількість машин	Фактична інтенсивність	Визначена інтенсивність	Відносна похибка визначення інтенсивності
Порівняння кадрів	3	468	547	156	182,33	16,88%
Модуль bioinspired	3	468	490	156	163,33	4,70%

Отримана при використанні модулю bioinspired, похибка 4,7% є прийнятною і свідчить про спроможність цього методу надавати якісну інформацію про інтенсивність транспортного потоку. Метод порівняння кадрів дає в 3,5 рази більшу похибку через меншу кількість пікселів рухомих об’єктів, що було отримано з порівняльних таблиць.

Таким чином, недоліки методів, заснованих на виявленні руху:

- Нерухомий транспорт не розглядається, але водночас він впливає на дорожню ситуацію, змушуючи водіїв інших транспортних засобів виїжджати на сусідню смугу руху в об'їзд.
- Неможливість виявлення транспорту в режимі реального часу через повільну швидкість роботи.

Тому для подолання описаних вище недоліків було вирішено використовувати нейронну мережу та використовувати метод, який показав себе найкращим чином (метод виявлення руху на основі біоінспірованого модуля OpenCV) для формування кластера даних для навчання цієї мережі.

Для того, щоб показати ефективність виявлення транспорту за допомогою нейронної мережі, було вирішено провести експеримент, в якому ми порівняли метод біоінспірації та метод нейронної мережі. На рис. 4.1 показано зображення з камери спостереження, розташованої у місті Києві. На рис.4.2представлений результат обробки зображення з застосуванням нейронної мережі.

У таблиці 4.2 наведено порівняння результатів використання методів виявлення дорожнього руху для задачі визначення інтенсивності транспортного потоку.

При оцінці безпеки дорожнього руху та в системах управління транспортним рухом часто використовують максимальне (пікове) або середньорічне значення інтенсивності. При цьому значення інтенсивності руху та показника завантаженості TLCR нерівномірне протягом доби, що і показано на рисунку 4.3. Дослідження проводилось в місті Києві на перехресті вулиць Юрія Іллєнка та Олени Теліги.

Рис. 4.1. Оригінальне зображення з камери

Рис. 4.2. Результат обробки зображення з застосуванням нейронної мережі

Таблиця 4.2. Результати ідентифікації інтенсивності руху

	Час, год.	Фактична кількість машин	Визначена кількість машин	Фактична інтенсивність	Визначена інтенсивність	Відносна похибка інтенсивності
Модуль bioinspired	3	817	873	272	291,00	6,85%
Нейронна мережа	3	817	851	272	283,66	4,16%

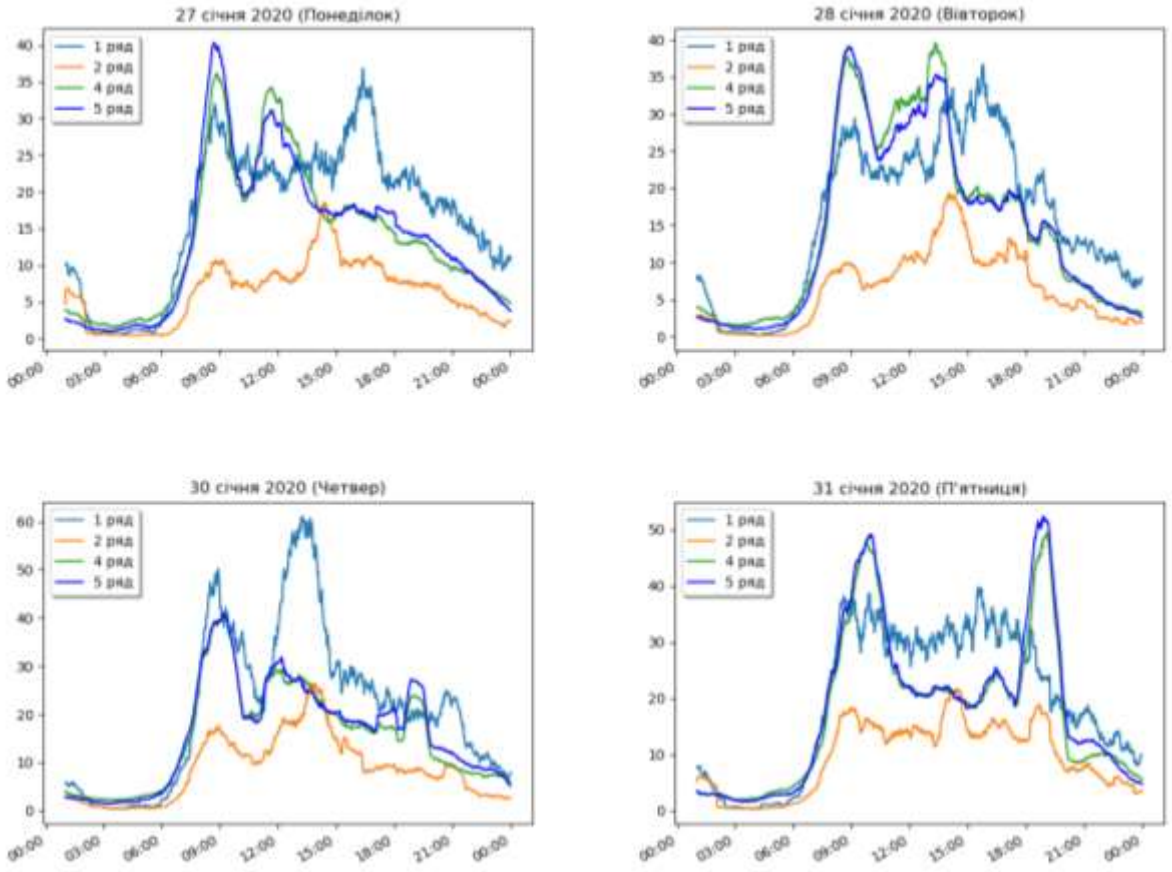


Рис 4.3. Динаміка змінювання показника завантаженості транспортної ділянки TLCR

На рисунку 4.4 наведено графічне порівняння фактичної та обчислюваної інтенсивності транспортного руху. Інтенсивність визначається за послідовними значеннями показника завантаженості (алгоритм описано в розділі 2.2.

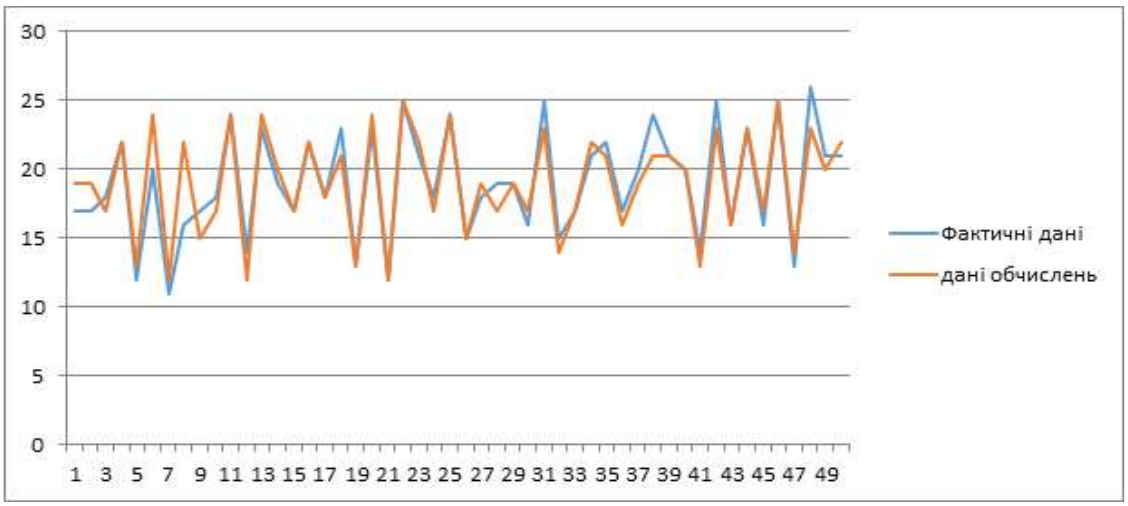


Рис. 4.4. Порівняння фактичних значень інтенсивності та значень, отриманих за даними відеоряду

4.2 Модель класифікації рівнів інтенсивності дорожнього руху при різній завантаженості

Для перевірки моделі класифікації рівнів інтенсивності дорожнього руху було проведено дослідження визначення значень TLCR розробленою інформаційною технологією для 1-но хвилинних відео з різною завантаженістю. На рис. 4.5-4.7 представлені результати визначення. На рис. 4.5 зображена динаміка змінювання показника завантаженості транспортної ділянки TLCR для відео з низькою завантаженістю, де видно відносно не високу кількість піків, які характеризують зафіксований транспортний засіб. На рис. 4.6 зображена динаміка змінювання показника завантаженості транспортної ділянки TLCR для відео з середньою завантаженістю. З рисунків видно, що для середньої завантаженості транспортної ділянки характерне збільшення числа піків на графіку (тобто числа автомобілів) в порівнянні з рис. 4.5, але ширина піків лишається такою ж, тобто швидкість руху не змінилась.

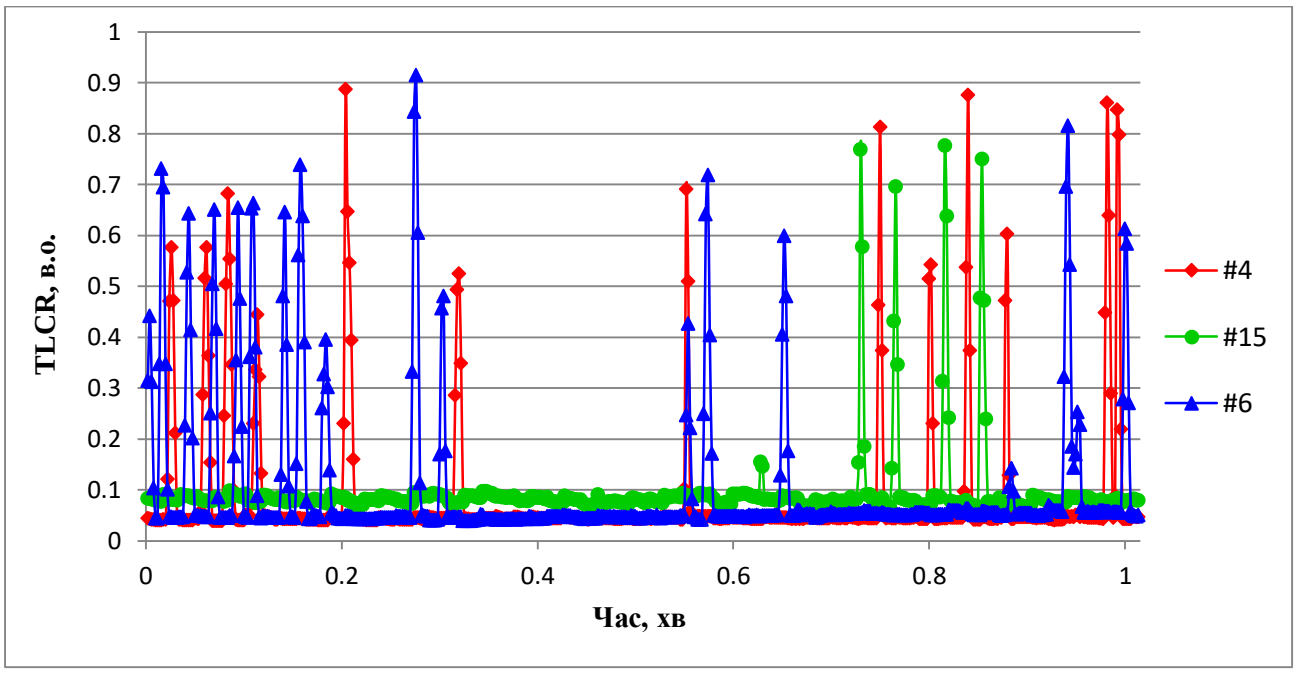


Рис. 4.5. Динаміка змінювання показника завантаженості TLCR для транспортної ділянки з низьким завантаженням транспортної ділянки для трьох різних відео.

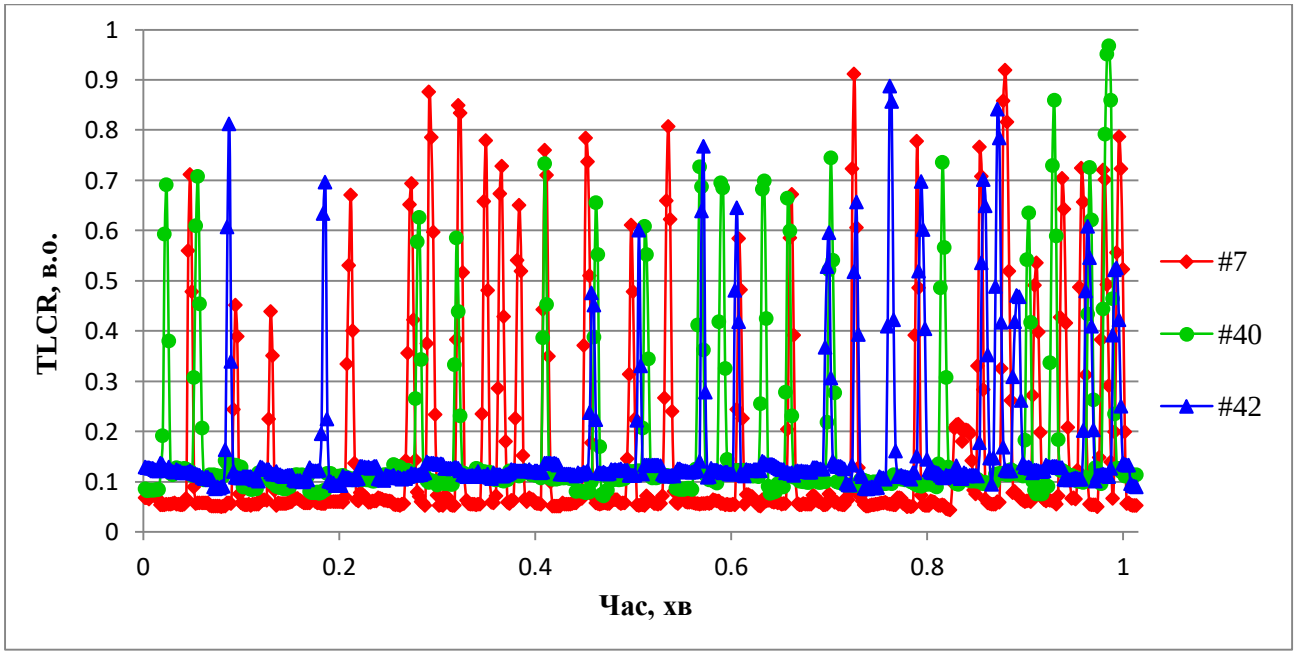


Рис. 4.6. Динаміка змінювання показника завантаженості TLCR для транспортної ділянки з середнім завантаженням транспортної ділянки для трьох різних відео.

На рис. 4.7 зображена динаміка змінювання показника завантаженості транспортної ділянки TLCR для відео з високою завантаженістю. Для високої завантаженості транспортної ділянки характерне також збільшення кількості піків (так само як і для середньої завантаженості) та збільшення ширини піків – що говорить про зниження швидкості руху автомобілів. Для відео №17 (рис. 4.7) видно плато на 0,8 сек, що говорить про повну зупинку транспортного засобу.

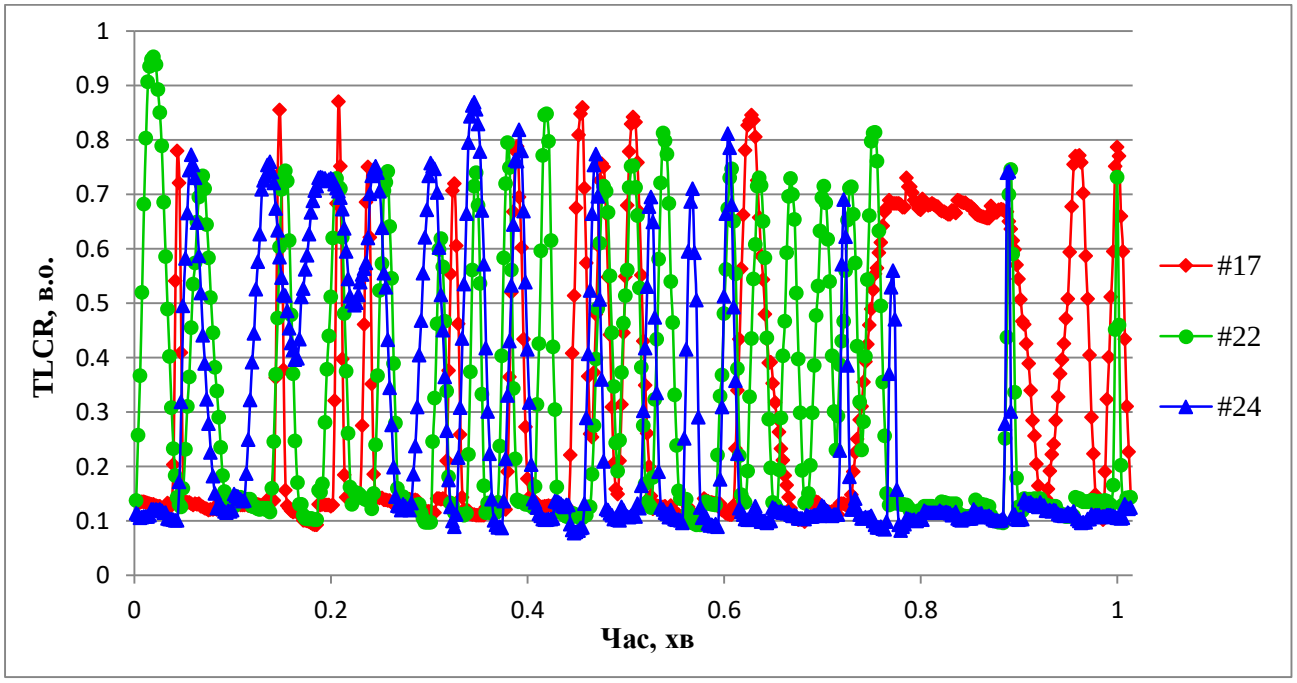


Рис. 4.7. Динаміка змінювання показника завантаженості TLCR для транспортної ділянки з високим завантаженням транспортної ділянки для трьох різних відео.

На рис. 4.8 зображено діаграму середніх за 1 хв значень TLCR для кожного відео з поділом на низьке, середнє та високе завантаження:

1) 1-но хвилинне TLCR в інтервалі від 0 до 0,11 – низьке завантаження,

2) 1-но хвилинне TLCR в інтервалі від 0,11 до 0,25 – середнє завантаження,

3) 1-но хвилинне TLCR в інтервалі від 0,25 до 0,35 – високе завантаження.

В таблиці 4.3. представлені дані визначених та фактичних значень транспортних засобів, що відповідає кількості піків на рис. 4.5-4.7.

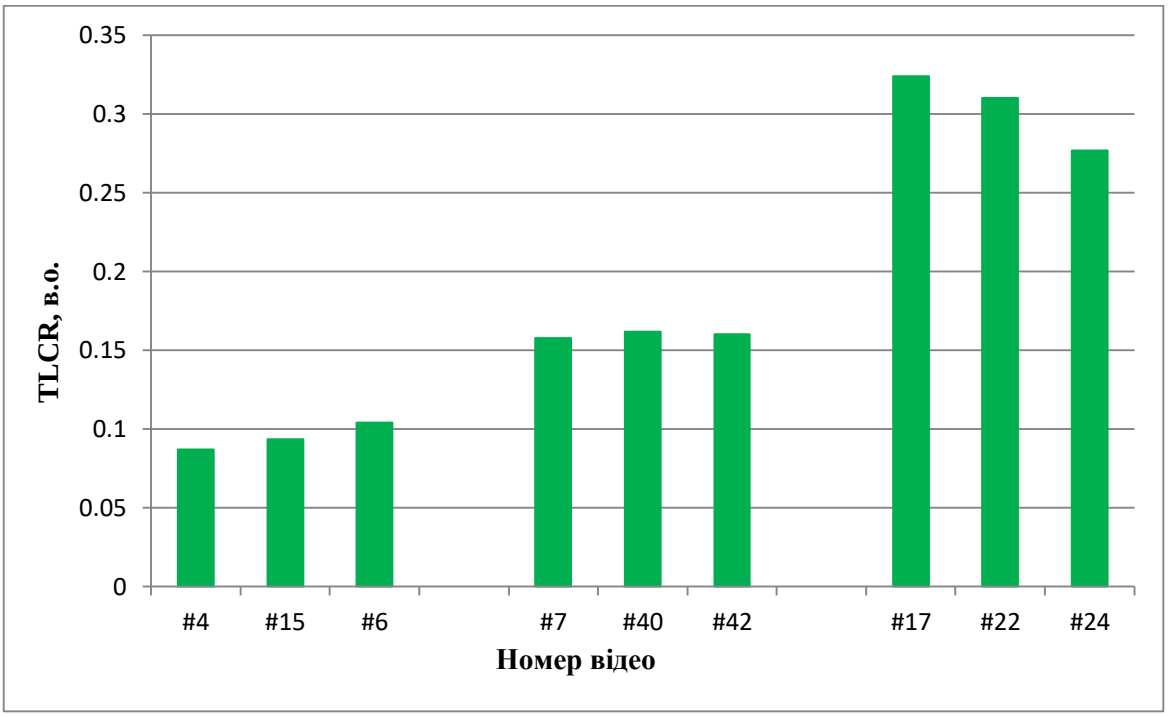


Рис. 4.8. Середні за одну хвилину значення показника завантаженості TLCR для різних відео

Таблиця 4.3. Результати ідентифікації інтенсивності руху

Номер відео	Фактична кількість автомобілів, од.	Визначена кількість автомобілів, од.
#4	13	13
#15	4	4
#6	17	16
#7	25	25
#40	17	17
#42	15	15
#17	13	13
#22	20	20
#24	13	14

Таким чином, модель визначення показника завантаженості транспортної ділянки TLCR може бути використана для класифікації рівня завантаженості транспортної ділянки.

4.3 Дослідження точності розробленої технології визначення інтенсивності дорожнього руху

В роботі [123], для оцінки середньої продуктивності відео-системи дорожнього руху запропоновано використання чотирьох показників: точність (Accuracy), повнота (Recall), влучність (Precision) та F-міра (F-measure). Точність (Accuracy) використовується для оцінки різниці між підрахованим значенням і справжнім значенням, яке можна визначити як:

$$A = 1 - \frac{|N_C - N_R|}{N_R} \tag{4.1}$$

деA– точність, N_R- спостережувана кількість транспортних засобів,

N_C– підрахована кількість.

Повнота (Recall) – це міра успішності методу у виявленні відповідних об’єктів із набору, тобто відсоток відповідних виявлених об’єктів у всіх відповідних об’єктах, тоді як влучність (Precision) – відсоток відповідних виявлених об’єктів у всіх виявлених об’єктах. F-міра – це середньозважене гармонічне середнє значення Recall та Precision, яке поєднує результати Recall та Precision. Три показники можна виразити як:

$$Recall = \frac{TP}{TP+FN}, \ \ Precision = \frac{TP}{TP+FP}, \tag{4.2}$$

$$F = \ 2 \cdot \frac{Recall \cdot Precision}{Recall+Precision} \tag{4.3}$$

де TP - кількість справжніх позитивних результатів (транспортних засобів, які були успішно підраховані),

FN - кількість помилкових негативів (транспортних засобів, які слід було б порахувати, але їх не було пораховано),

FP - кількість помилкових позитивних результатів (помилкових об'єктів, які зараховувались як транспортні засоби).

Використовуючи TP, FN і FP, N_R, N_C, точність можна також виразити як:

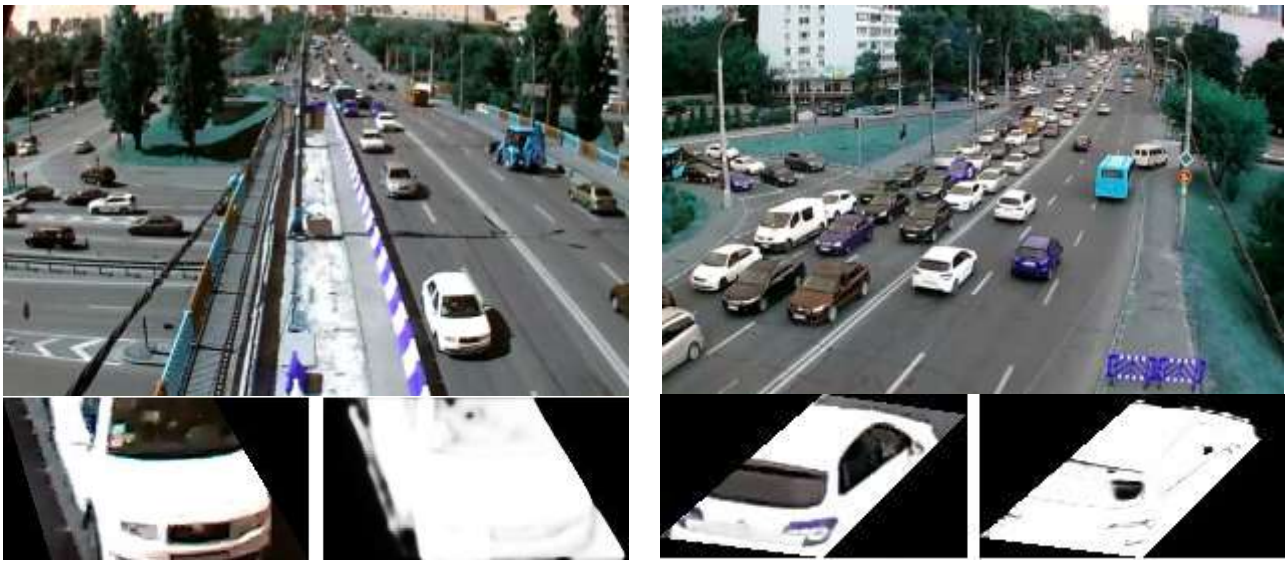
$$N_R = TP + FN, N_C = TP + FP \tag{4.4}$$

$$A = 1 - \frac{|FN - FP|}{TP+FN} \tag{4.5}$$

Згідно [123], точність (Accuracy) може оцінити лише загальну різницю між підрахованими значеннями та справжніми значеннями, і не може відображати помилки трактування шуму як транспортного засобу або помилки виявлення відсутніх. Гірше, результат точності (Accuracy) може наблизитися до 1,0, коли кількість цих двох помилок близька або навіть дорівнює. Отже, всебічний аналіз точності та F-міри може бути більш науковим для оцінки ефективності методу.

Результати досліджень показані на рис. 4.9 та в таблицях 4.4-4.7.

а) б)



в) г)

Рис. 4.9. Результати експериментів: а) вул. О.Теліги; б) міст Г. Фукса; в) Повітрофлотський міст, камера №1; г) Повітрофлотський міст, камера №2.

Таблиця 4.4. Результати обчислень для вул. О.Теліги

Lane No.	***True***	***Counted***	***TP***	***FN***	***FP***	***Recall***	***Precision***	***F-measure***	***Accuracy***
Lane 1	207	207	207	0	0	1,0	1,0	1,0	1,0
Lane 2	161	161	161	0	0	1,0	1,0	1,0	1,0
Total	368	368	368	0	0	1,0	1,0	1,0	1,0

Таблиця 4.5. Результати обчислень для мосту Г. Фукса

Lane No.	***True***	***Counted***	***TP***	***FN***	***FP***	***Recall***	***Precision***	***F-measure***	***Accuracy***
Lane 1	202	200	200	2	0	0,9901	1,0	0,9950	0,9901
Lane 2	223	219	219	4	0	0,9821	1,0	0,9910	0,9821
Total	425	419	419	6	0	0,9859	1,0	0,9929	0,9859

Таблиця 4.6. Результати обчислень для Повітрофлотського мосту, камера №1

Lane No.	***True***	***Counted***	***TP***	***FN***	***FP***	***Recall***	***Precision***	***F-measure***	***Accuracy***
Lane 1	266	265	265	1	0	0,9962	1,0	0,9981	0,9962
Lane 2	303	301	301	2	0	0,9934	1,0	0,9967	0,9934
Total	569	566	566	3	0	0,9947	1,0	0,9974	0,9947

Таблиця 4.7. Результати обчислень для Повітрофлотського мосту, камера №2

Lane No.	***True***	***Counted***	***TP***	***FN***	***FP***	***Recall***	***Precision***	***F-measure***	***Accuracy***
Lane 1	170	169	169	1	0	0,9941	1,0	0,9971	0,9941

Всі чотири показники продуктивності для камери на вул. О.Теліги дорівнюють 1,0, що означає що всі транспортні засоби були правильно визначені (див. рис. 4.9а, табл.4.4).

Для камери на мості Г. Фукса значення F-міри дорівнює 0,9929, а точності – 0,9859; серед 419 правильно визначених цілей було пропущено 6 об’єктів (див. рис. 4.9 б, табл.4.5). Для камери на Повітрофлотському мості (камера №1) значення F-міри дорівнює 0,9974, а точності – 0,9947; серед 569

правильно визначених цілей було пропущено 3 об’єкти (див. рис. 4.9 в, табл.4.6). Для камери на Повітрофлотському мості (камера №2) було пропущено 1 об’єкт (див. рис. 4.9 г, табл.4.7).

В усіх випадках помилки відбувались через те, що авто проїжджало між смугами і тому в кадр потрапляло відразу два авто. В такому випадку показник TLCR не падає нижче свого порогового значення і система вважає, що попереднє авто ще не проїхало, і тому транспортний засіб помилково не зараховується (рис. 4.10).

а)  б) 

Рис. 4.10. Приклади помилково не зарахованих транспортних засобів: а) Повітрофлотський міст, камера №1; б) Повітрофлотський міст, камера №2.

Порівняємо базові показники запропонованої системи з подібними методами, які представлені в роботах [123-127]. Отримані результати порівняння представлені в табл. 4.8. При співставних кількостях проаналізованих транспортних засобів було отримано, що точність (Accuracy) є вищою у Chen, Y. [123], хоча загальна кількість помилок 21, з них 10 пропущених транспортних засобів та 11 помилково порахованих. Таким чином, ці значення у формулі (3) компенсуються і в результаті буде враховуватись тільки одне помилкове значення. Тоді як показник F-міра, який враховує пропущені та помилково пораховані значення, є більш надійним для визначення точності системи. У Chen, Y. [123] F-міра є нижчою.

Загалом в експерименті було правильно визначено 1522 транспортних засобів та пропущено 10 транспортних засобів. Єдиною причиною пропущених

авто є рух між смугами автостради. В роботі [123] також згадуються погане освітлення та рух великих та довгих автомобілів, які можуть закривати менші авто, як причини, що спричиняють помилки. В нашому дослідженні таких причин не було виявлено, хоча це може бути об’єктом майбутніх досліджень. Отримані значення F-міра – 0.9967 та точність (Accuracy) – 0.9935 демонструють високу надійність визначення інтенсивності дорожнього руху розробленої системи.

Таблиця 4.8. Порівняння експериментальних результатів із існуючими сучасними методами

Counting method	***True***	***Counted***	***TP***	***FN***	***FP***	***Recall***	***Precision***	***F-measure***	***Accuracy***
Bouvié, C. [124]	42	N/A	N/A	9	0	N/A	N/A	0,88	0,7857
Quesada, J. [125]	42	N/A	N/A	3	0	N/A	N/A	0,963	0,9286
Yang, H. [126]	58	N/A	N/A	7	0	N/A	N/A	0,9425	0,878
Abdelwahab, M. [127]	42	N/A	N/A	3	0	N/A	N/A	0,98215	0,96295
Chen, Y. [123]	1394	1395	1384	10	11	0,9982	0,9921	0,9925	0,9993
Запропоно-ванийметод	1532	1522	1522	10	0	0,9935	1,0	0,9967	0,9935

4.4 Дослідження прогнозування показника завантаженості TLCR

Розроблену модель прогнозування, яка була описана в розділі 3.3, перевірялась на 2-х тижневих відео-записах дорожнього руху отриманих з камери, яка встановлена біля метро Дорогожичі в місті Києві. Перший тиждень відео-записів було використано для навчання рекурентної нейронної мережі LSTM. А другий тиждень (п’ять днів – з суботи до середи) використали для оцінки точності прогнозування транспортної системи. Визначали

прогнозований та реальний показник TLCR з 5-ти, 10-ти та 15-ти хвилинним усередненням для п’яти днів.

Отримані результати наведені в таблицях 1-3 (Додаток В) та зображені на графіках ефекту прогнозування потоку руху за показником TLCR для різних днів (рис. 1-15, Додаток Г). На рис. 1-15 Додатку Г, вісь X представляє 13 годин світлої частини доби, коли обсяг трафіку потрібно передбачити, а вісь Y відображає показник завантаженості TLCR.

З рис. 1-15 Додатку Г можна легко зробити висновок, що отриманий прогноз про рух автомобілів має достатньо високу точність. Для вихідних днів – суботи та неділі (рис. 4.11 та 4.12) показник TLCR виявився найменшим – це можна пояснити низькою активністю на дорозі у ці дні. У будні дні (рис. 4.13-4.15) спостерігається збільшення загального показника TLCR у ранковий та вечірній час, що відповідає найбільшій активності на дорогах.

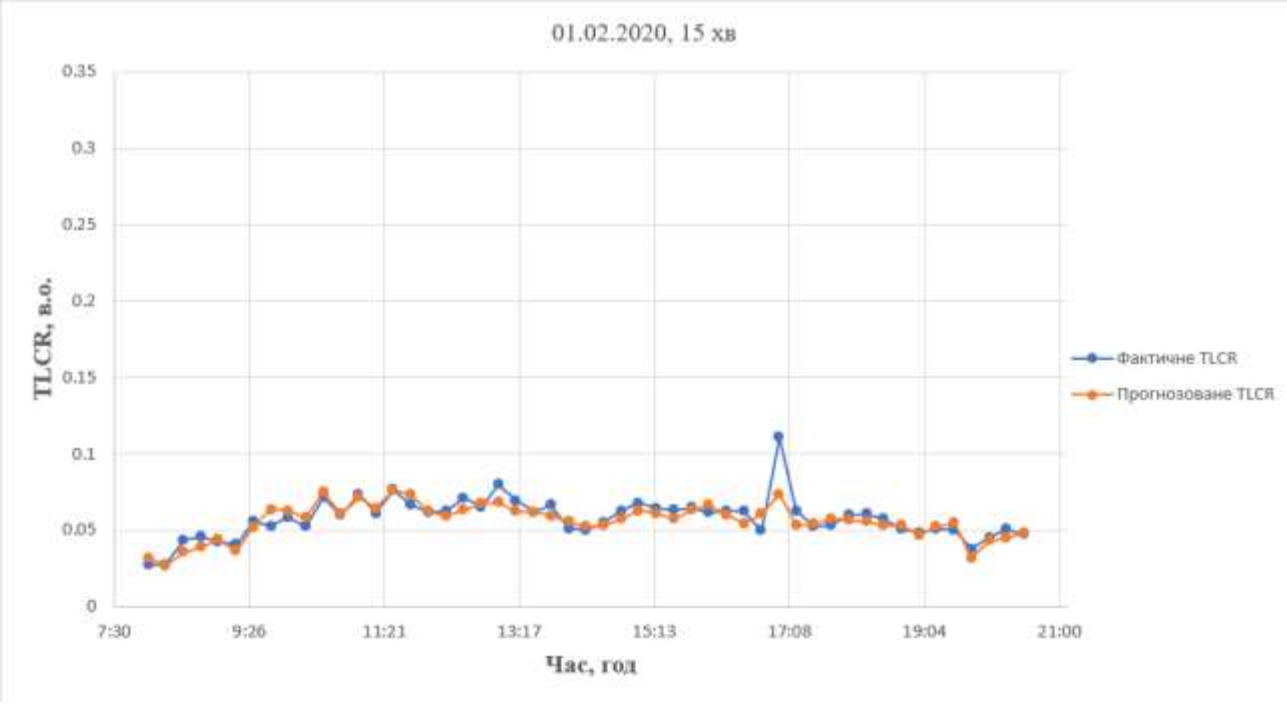


Рис. 4.11. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у суботу з 15-ти хвилинним усередненням

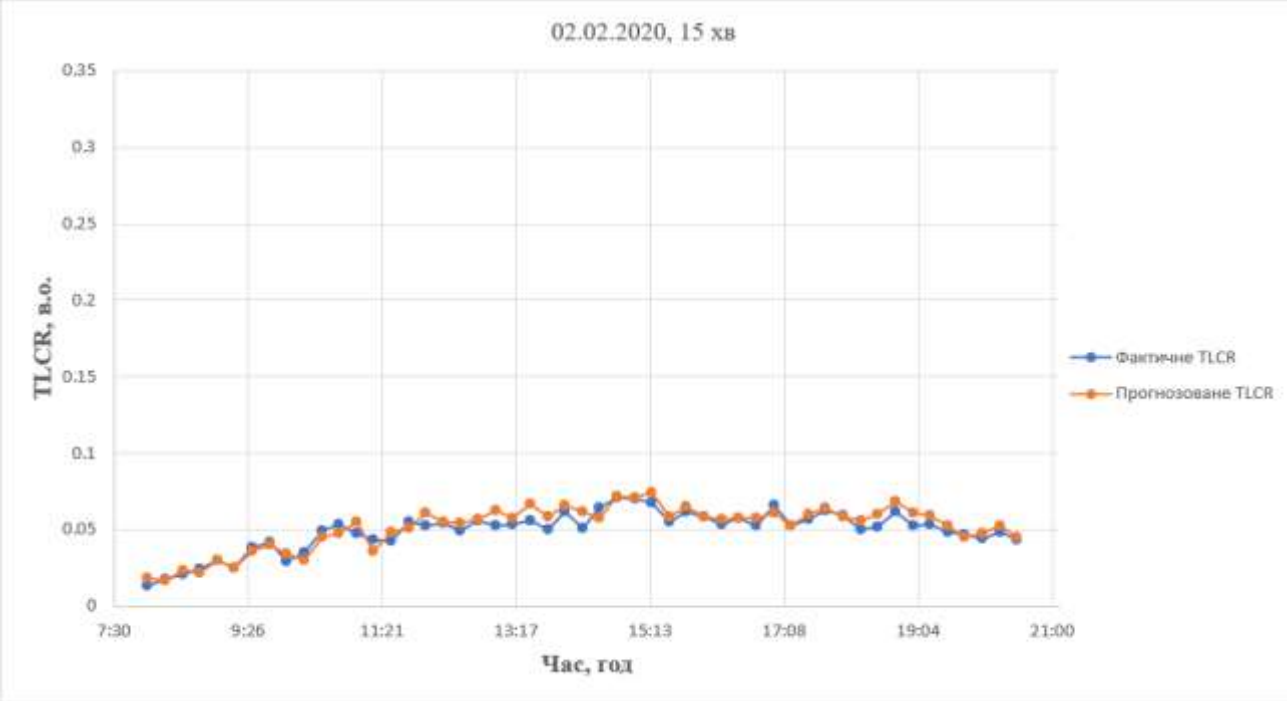


Рис. 4.12. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у неділю з 15-ти хвилинним усередненням

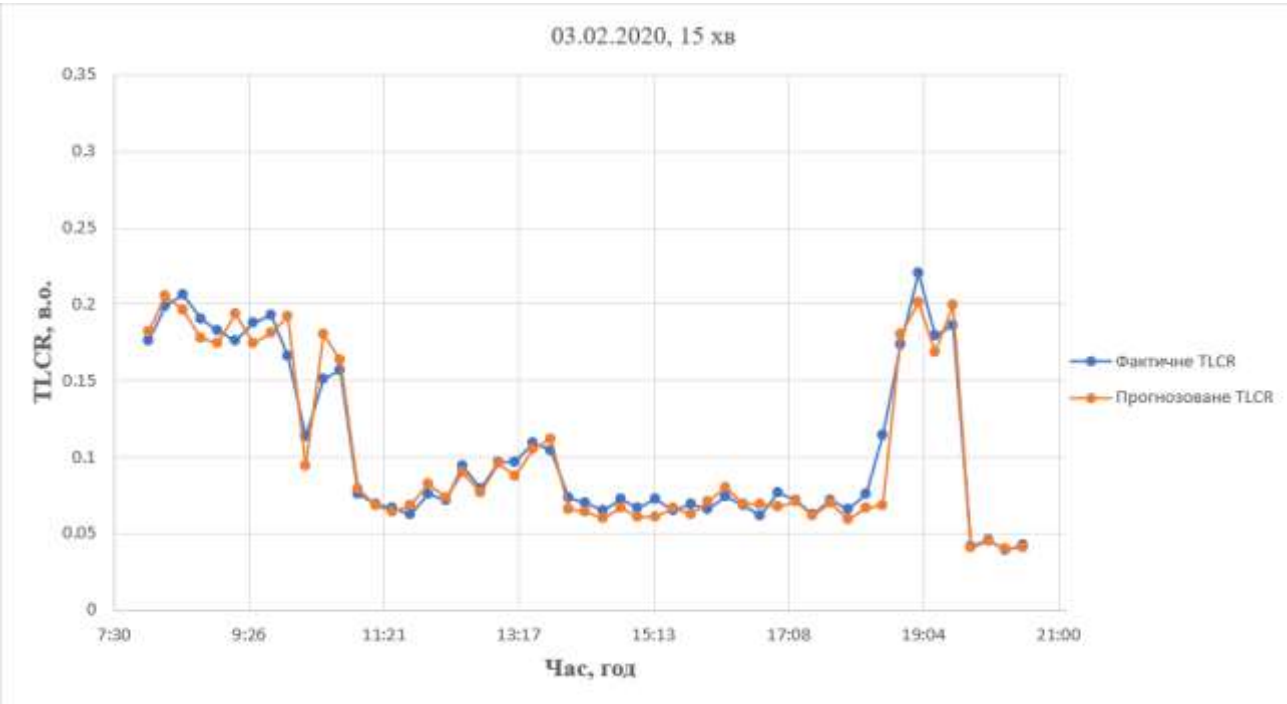


Рис. 4.13. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) в понеділок з 15-ти хвилинним усередненням

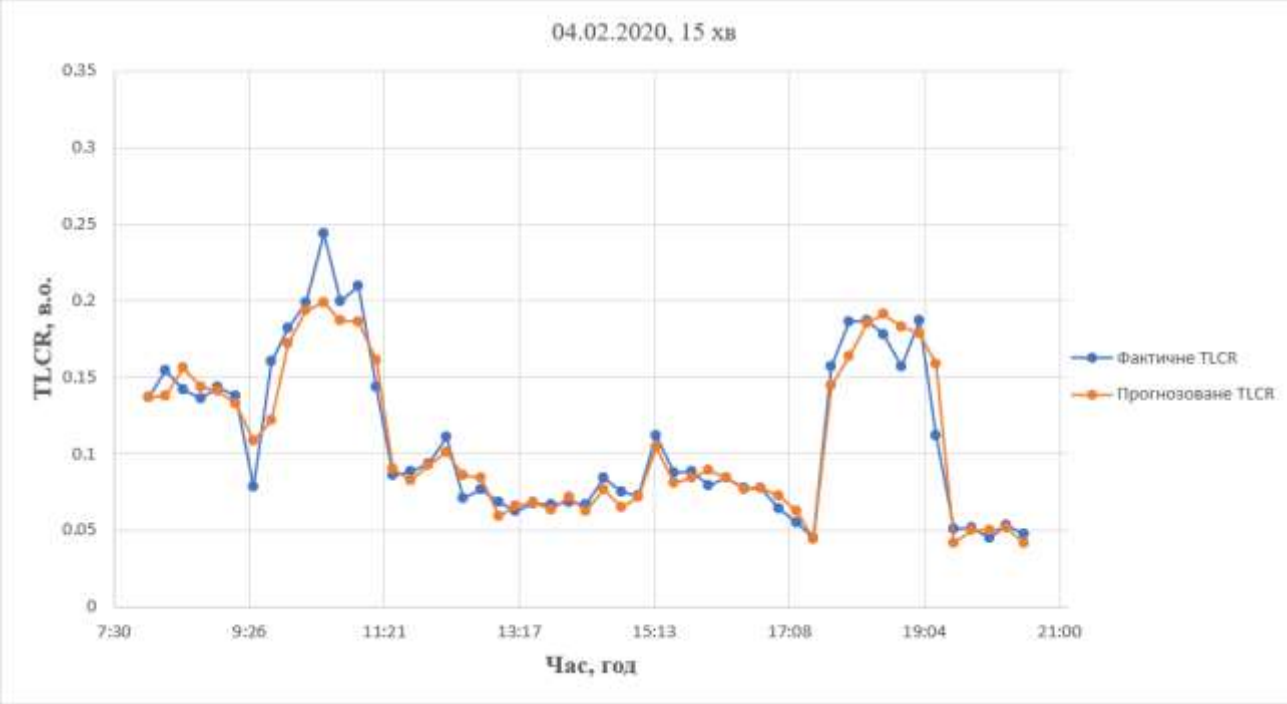


Рис. 4.14. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у вівторок з 15-ти хвилинним усередненням

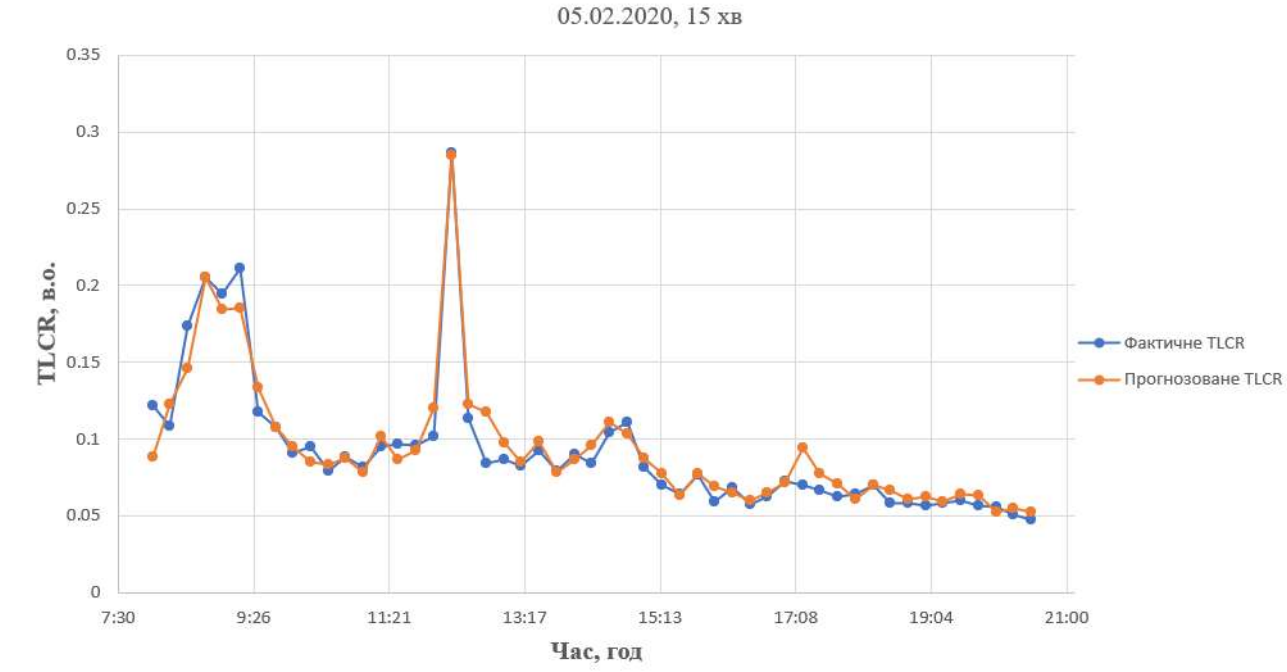


Рис. 4.15. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) в середу з 15-ти хвилинним усередненням

4.4.1 Визначення абсолютної похибки прогнозування показника завантаженості TLCR

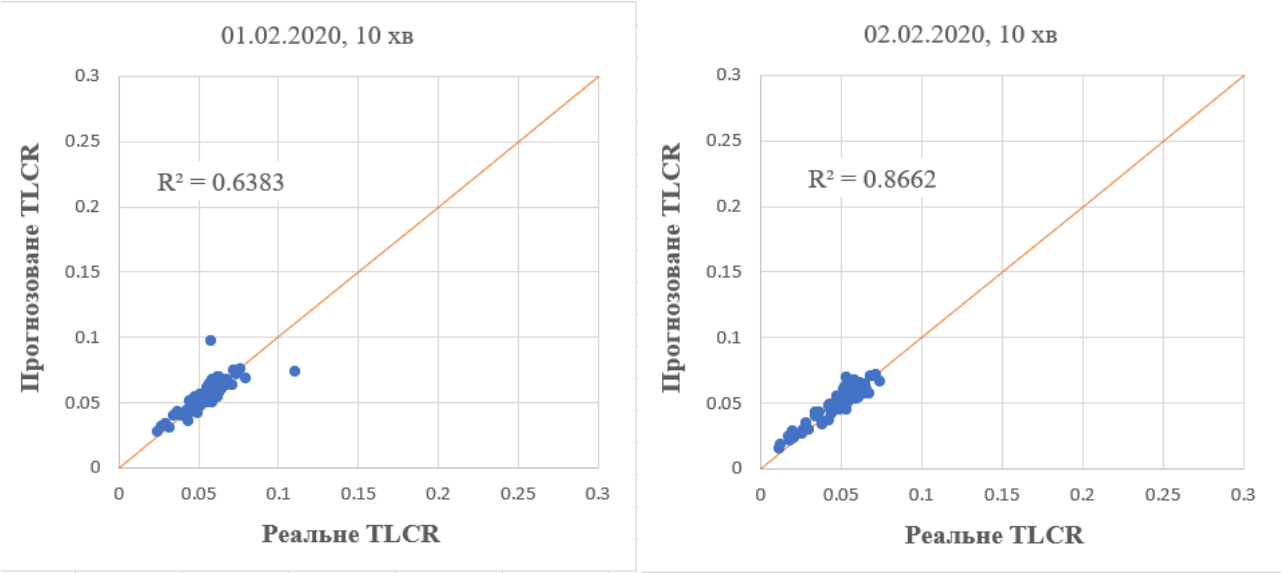
Графіки абсолютної похибки прогнозування показника завантаженості TLCR для різних днів тижня зображено на рис. 1-15 Додатку Д, де вісь X представляє реальне значення показника завантаженості TLCR, а вісь Y - прогнозоване значення показника завантаженості TLCR.

Отримані значення коефіцієнта R^2 для кожного дня з 5-ти, 10-ти та 15-ти хвилинним усередненням занесені до таблиці 4.9. Отримані результати демонструють кращу точність для показника TLCR з 15-ти хвилинним усередненням (середнє R^2 = 0.8852) та найгіршою точністю для показника TLCR з 10-ти хвилинним усередненням (середнє R^2 = 0.83056). Також можна відмітити нерівномірність показника точності для різних днів тижня: низькі значення для вихідних днів (1.02 – субота, R^2 від 0.6383 до 0.7138; рис. 4.16), і високі – для будніх днів (3.02 – понеділок, R^2 від 0.9284 до 0.9561; рис. 4.17).

Отримані значення R^2 для прогнозування співставні з аналогічною роботою [128] і демонструють хороший результат отриманого прогнозування.

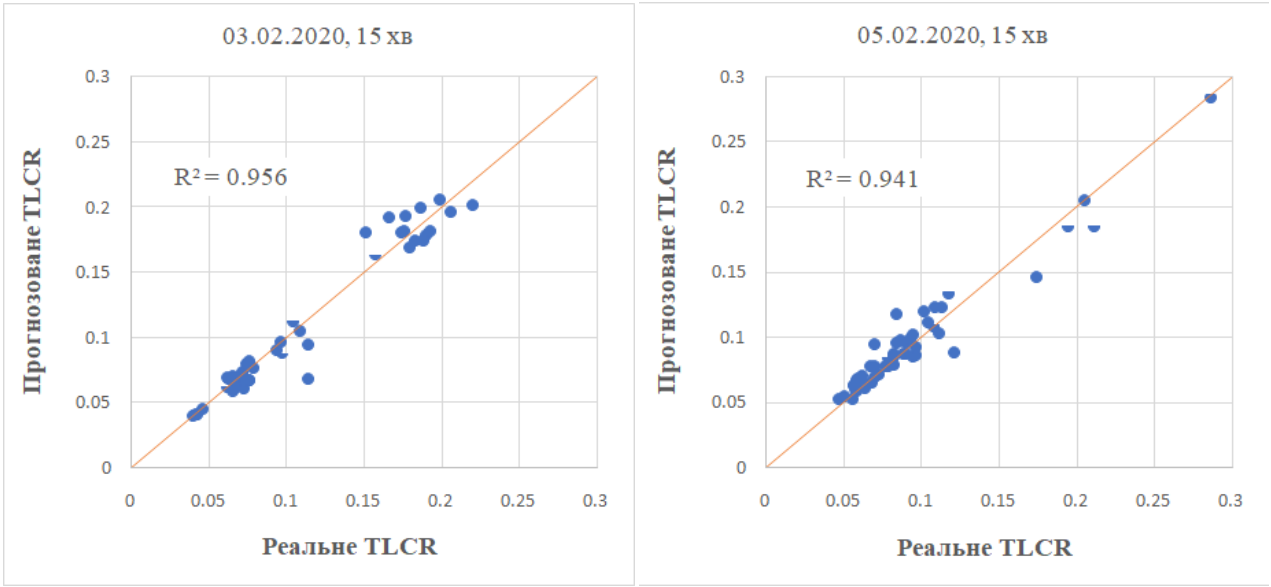
Таблиця 4.9. Коефіцієнт кореляції R^2 між фактичним та прогнозованим значенням TLCR для різних днів з різним усередненням

	R^2		
День	**TLCR (5 хв)**	**TLCR (10 хв)**	**TLCR (15 хв)**
1.02	0.682	0.6383	0.7138
2.02	0.8752	0.8662	0.895
3.02	0.9284	0.9311	0.9561
4.02	0.9283	0.9138	0.9195
5.02	0.867	0.8034	0.9416
Average	**0.85618**	**0.83056**	**0.8852**



а) б)

Рис. 4.16. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для суботи (а) та неділі (б) з 10-ти хвилинним усередненням



а) б)

Рис. 4.17. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для понеділка (а) та середи (б) з 15-ти хвилинним усередненням

4.4.2 Визначення точності прогнозування показника завантаженості TLCR

У цьому дослідженні для оцінки ефективності моделі прогнозування показника завантаженості TLCR було обрано найпоширеніші показники оцінки [129]: F-measure та Accuracy, які визначаються за формулами (4.3) та (4.5) відповідно, які були використані в розділі 4.3. F-measure– міра точності тесту, визначається як середньозважене повнота (recall) та влучність (precision). Точність (accuracy) є одним із показників ефективності класифікацій, який визначається як відношення правильної вибірки до загальної кількості проб.

На рис. 4.18-4.20 представлено динаміку зміни F-measure для прогнозованих значень показника завантаженості TLCR для різних днів тижня з 5-ти, 10-ти та 15-ти хвилинним усередненням. На рис. 4.21-4.23 представлено динаміку зміни accuracy для прогнозованих значень показника завантаженості TLCR для різних днів тижня з 5-ти, 10-ти та 15-ти хвилинним усередненням. Середні значення F-measure та Accuracy занесені до табл. 4.9 та зображені на рис. 4.24-4.25. Результати демонструють високу точність для прогнозування показника завантаженості TLCR з 15-ти хвилинним усередненням та низькі результати для 10-ти хвилинного усереднення показника TLCR, подібні результаті отримані в розділі 4.4.1. Загальна точність співставна з аналогічною роботою [129] і демонструють хороший результат отриманого прогнозування.

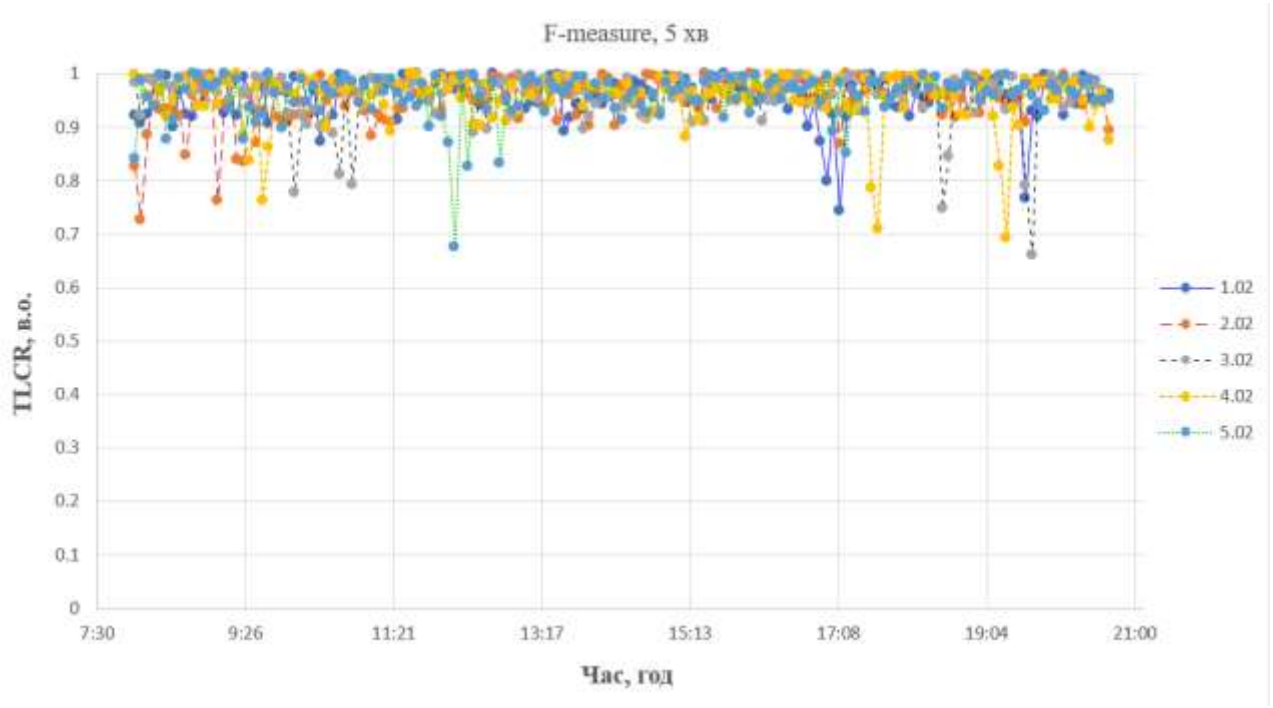


Рис. 4.18. Динаміка змін F-measure для різних днів з 5-ти хвилинним усередненням

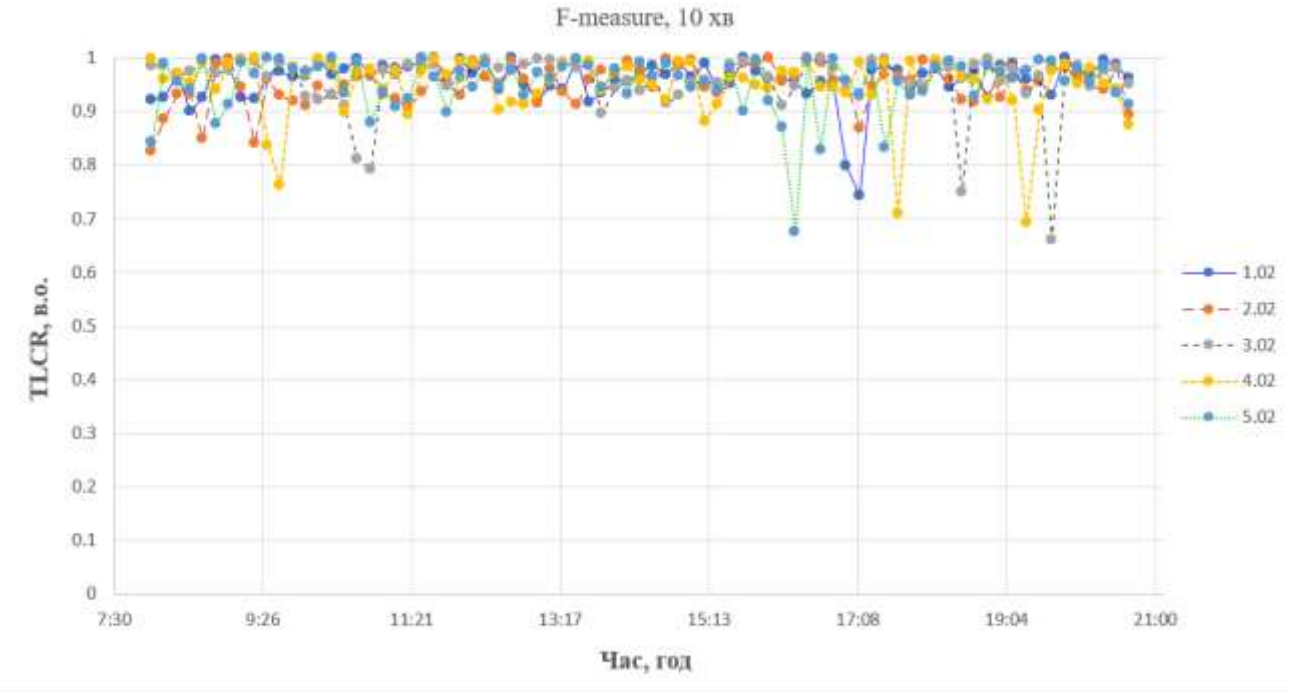


Рис. 4.19. Динаміка змін F-measure для різних днів з 10-ти хвилинним усередненням

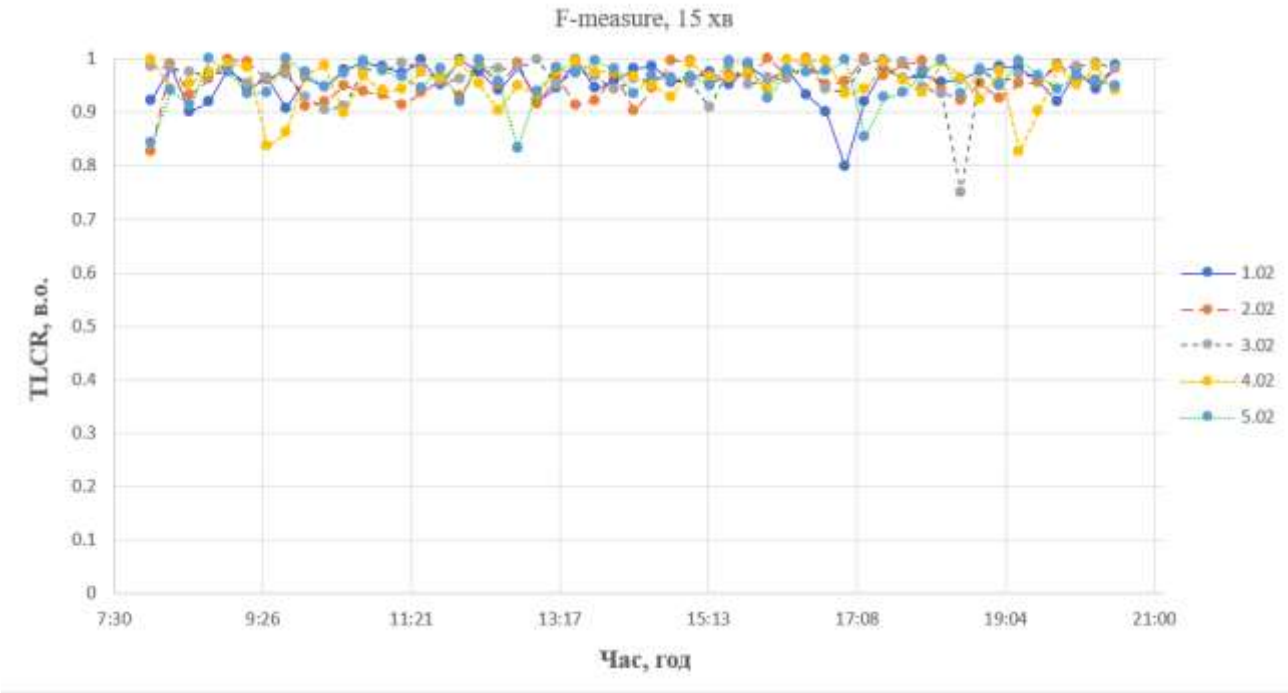


Рис. 4.20. Динаміка змін F-measure для різних днів з 15-ти хвилинним усередненням

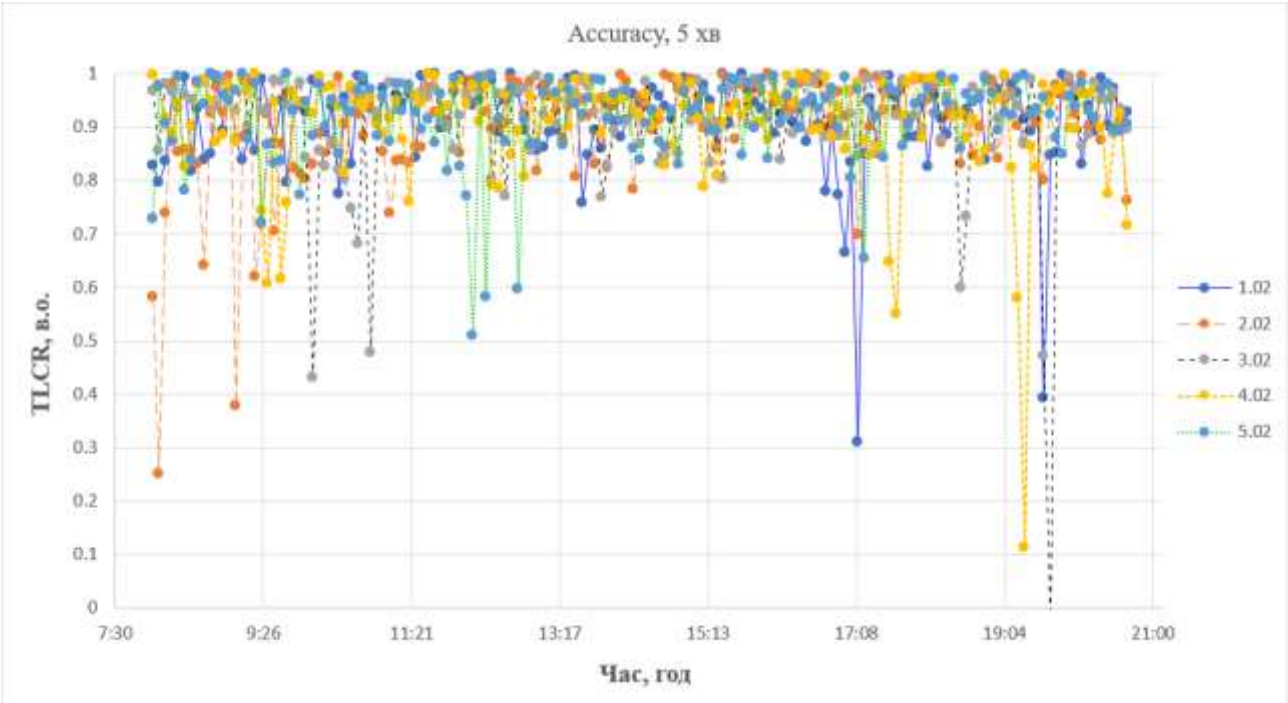


Рис. 4.21. Динаміка змін Accuracy для різних днів з 5-ти хвилинним усередненням

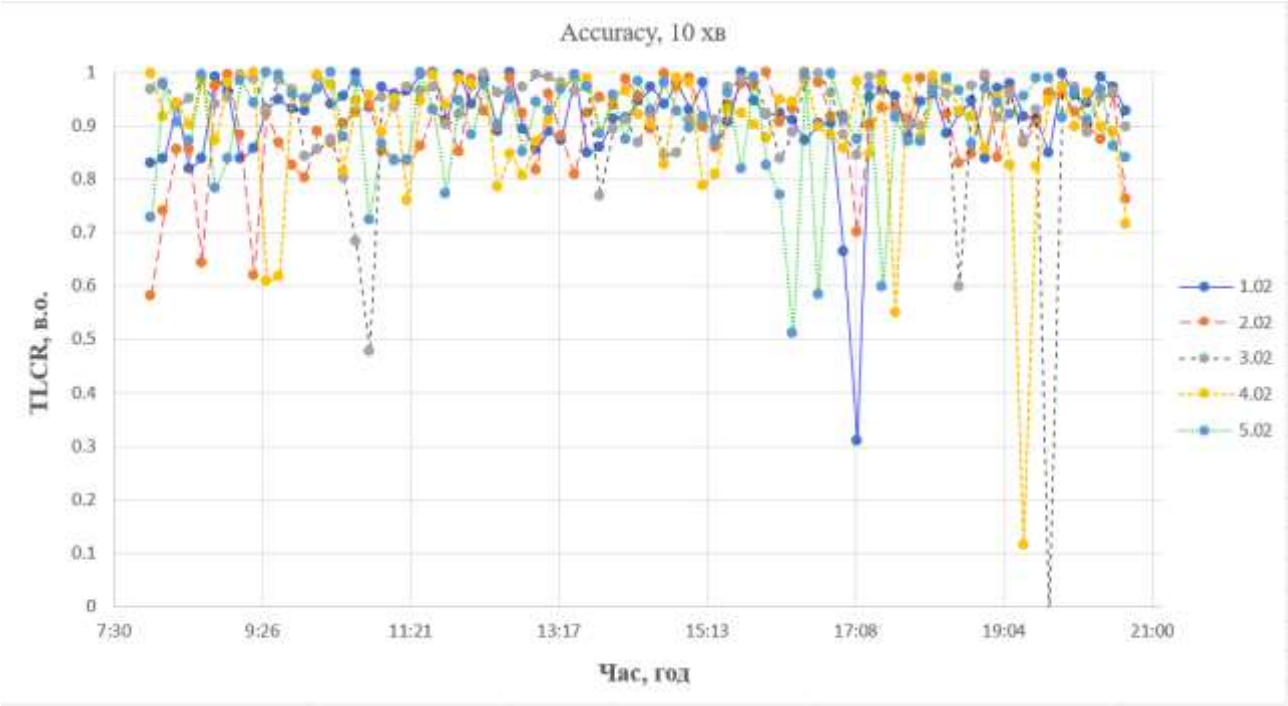


Рис. 4.22. Динаміка змін Accuracy для різних днів з 10-ти хвилинним усередненням

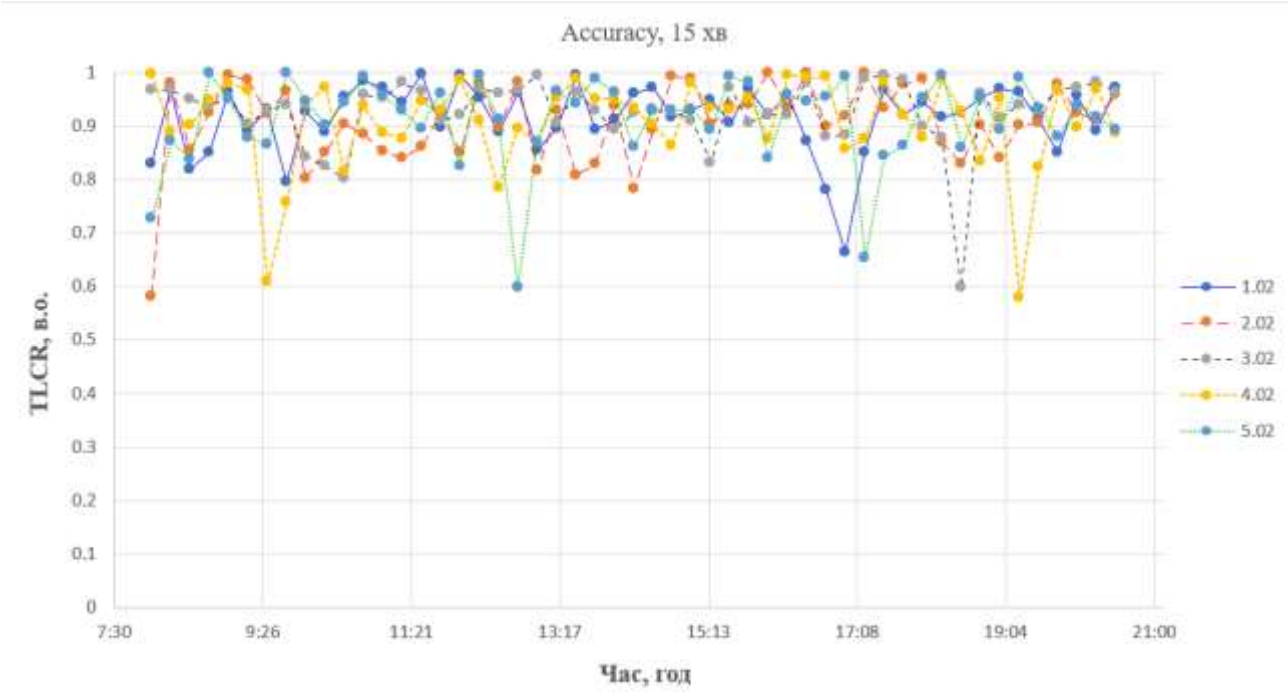


Рис. 4.23. Динаміка змін Accuracy для різних днів з 15-ти хвилинним усередненням

Таблиця 4.9. Точність F-measure та Accuracy прогнозування TLCR для різних днів з різним усередненням

	F-measure			Accuracy		
	5 хв	**10 хв**	**15 хв**	**5 хв**	**10 хв**	**15 хв**
1.02	0.957179	0.959739	0.957212	0.912205	0.917419	0.916647
2.02	0.954004	0.951087	0.955851	0.902587	0.897239	0.907401
3.02	0.957035	0.956948	0.961946	0.910784	0.909763	0.926601
4.02	0.954631	0.949549	0.955003	0.907642	0.896669	0.908511
5.02	0.958571	0.956871	0.957446	0.917044	0.914839	0.911586
Average	**0.956284**	**0.954839**	**0.957492**	**0.910052**	**0.907186**	**0.914149**

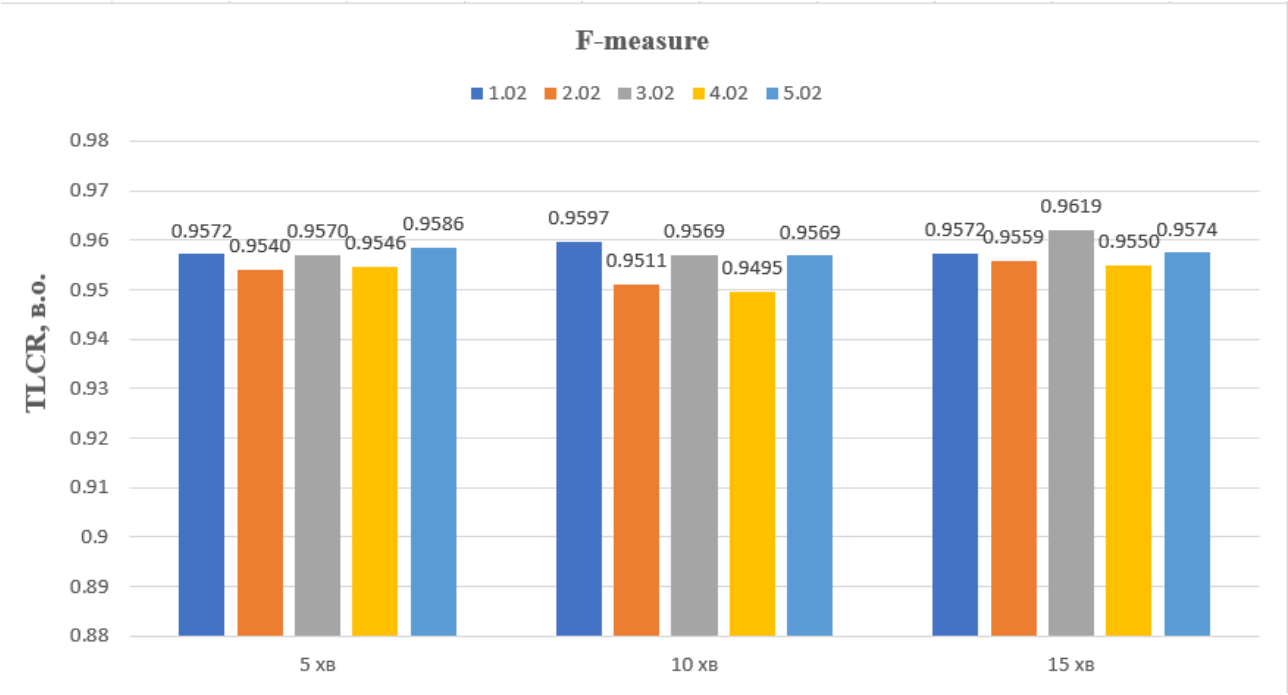


Рис. 4.24. Точність F-measure прогнозування TLCR для різних днів з різним усередненням

На рис. 4.26-4.28 зображена динаміка змін точності прогнозування показника TLCR і отриманих прогнозованих та реальних значень показника TLCR на яких можна спостерігати зниження значення точності при переході з низьких значень показника TLCR до високих і навпаки. Виявлена особливість є недоліком розробленої транспортної системи, яка буде вирішена в подальших дослідженнях. Також дана модель прогнозування буде порівнюватись з іншими існуючими в подальших дослідженнях.

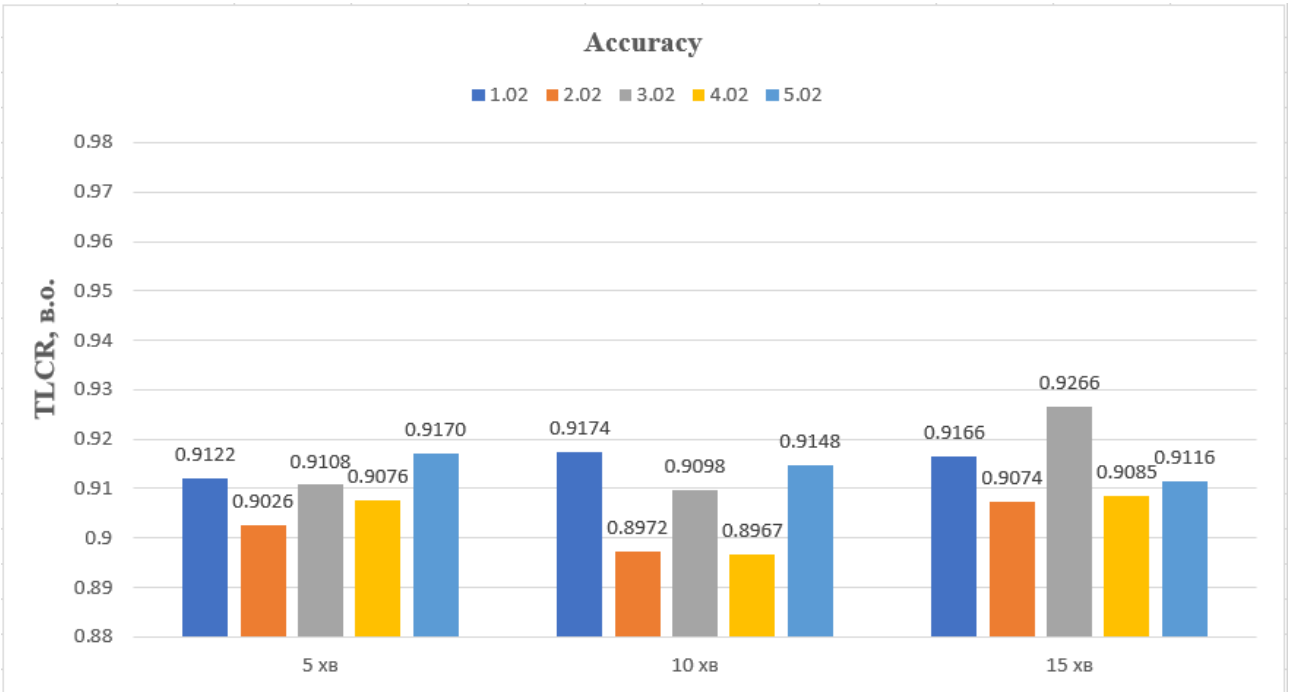


Рис. 4.25. Точність Accuracy прогнозування TLCR для різних днів з різним усередненням

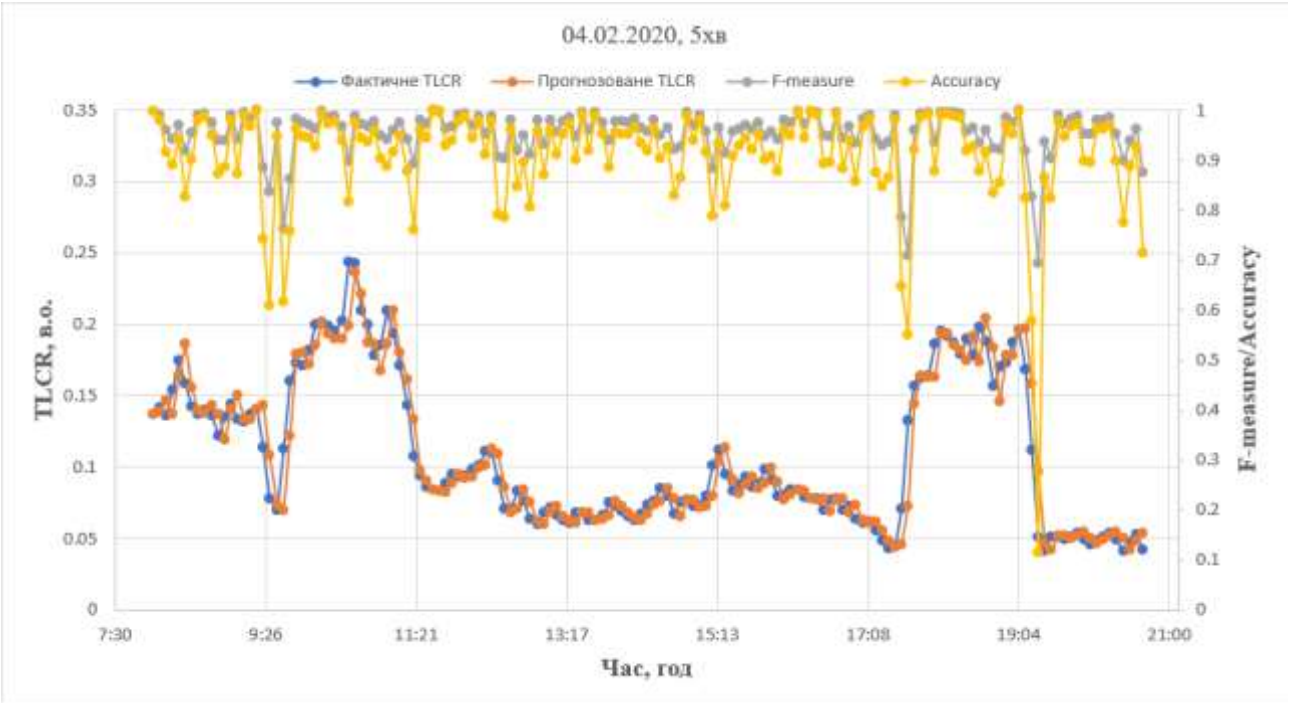


Рис. 4.26. Динаміка змін точності прогнозування TLCR і отриманих прогнозованих та реальних значень TLCR з 5-ти хвилинним усередненням

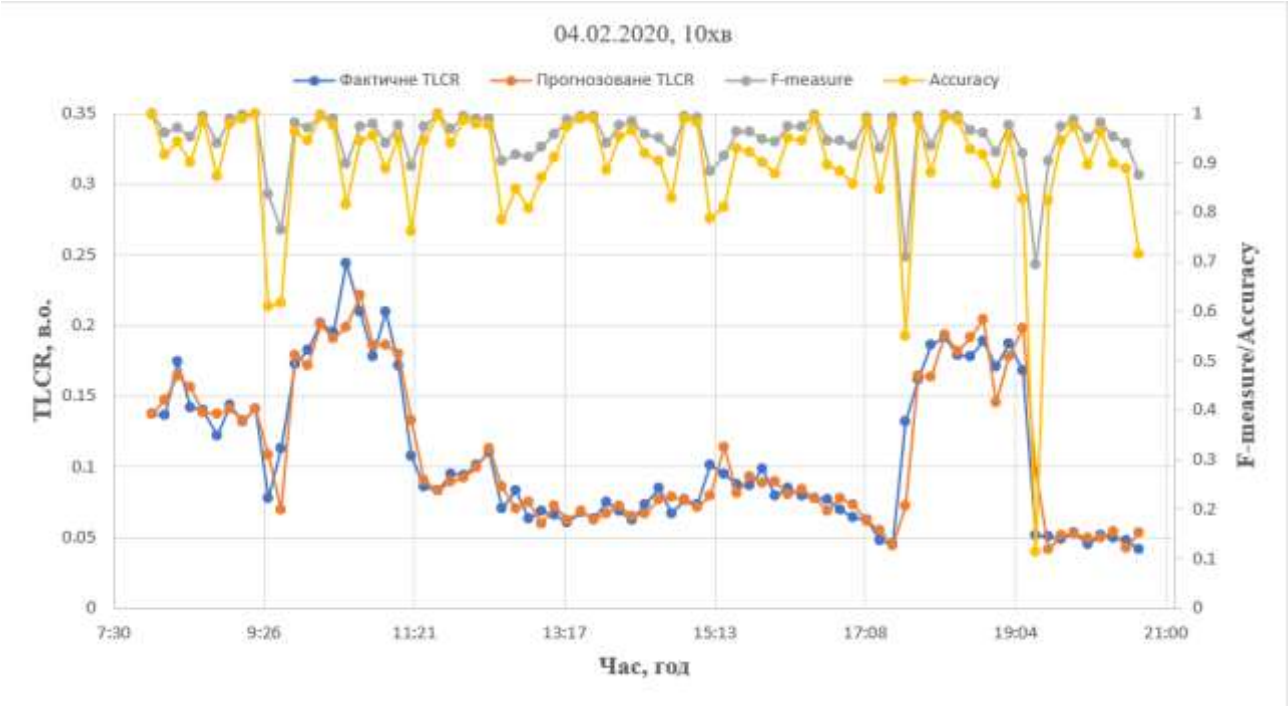


Рис. 4.27. Динаміка змін точності прогнозування TLCR і отриманих прогнозованих та реальних значень TLCR з 10-ти хвилинним усередненням

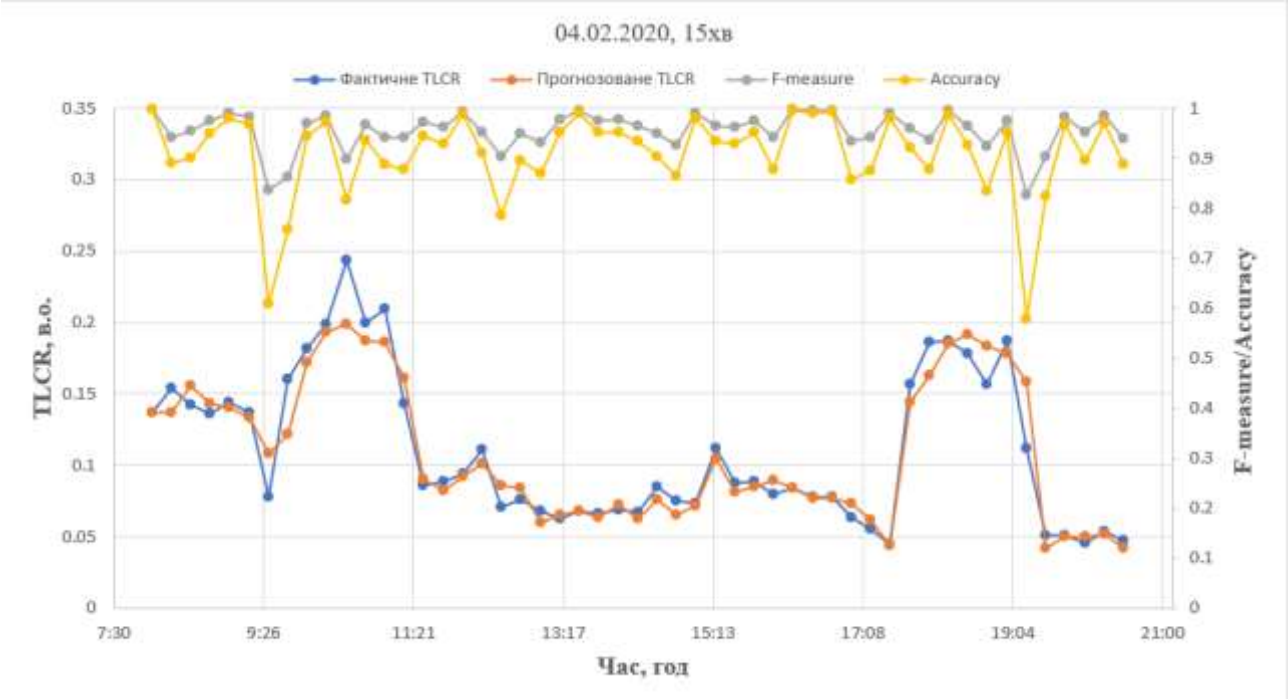


Рис. 4.28. Динаміка змін точності прогнозування TLCR і отриманих прогнозованих та реальних значень TLCR з 15-ти хвилинним усередненням

4.5 Висновки

Розроблено алгоритм оцінювання стану дорожнього руху за показником завантаженості транспортної ділянки TLCR отриманих з зображень отриманих з відеокамер, встановлених у різних місцях міста. Алгоритм дозволяє класифікувати затори за трьома рівнями завантаженості: низька, середня та висока завантаженість. Розроблений алгоритм досліджували на експериментальних даних записів з дорожніх відеокамер та було отримано задовільні результати виявлення та класифікації рівнів завантаженості.

Послідовність обробки та перетворень даних складають нову технологію визначення інтенсивності дорожнього руху, що забезпечує високу точність оцінки інтенсивності руху транспортних засобів на ділянці дорожнього руху. Розроблена технологія може працювати в умовах нестаціонарності параметрів транспортного руху. Завдяки використанню сегментації, замість класифікації, а також специфічного набору даних для навчання, технологія позбавлена таких недоліків як неправильно підібраний ракурс та відсутність транспортного засобу в існуючих базах даних для навчання. Запропонована система успішно підрахувала транспортні засоби з високою точністю, – середні значення F-міра та точність (Accuracy) досягли 0,9967 та 0,9935 відповідно.

Досліджено точність розробленої інформаційної системи довгострокового прогнозування показника завантаженості транспортної ділянки TLCR, яка базується на моделі навчання з рекурентною нейронною мережею LSTM. Отримана експериментальна середня точність 0,914 для п’яти днів (два вихідних та три робочих дні) демонструє високу ефективність результатів прогнозування.

ВИСНОВКИ

В дисертаційній роботі розв’язано актуальне наукове завдання підвищення точності визначення інтенсивності транспортного руху на основі аналізу даних відеопотоку в режимі реального часу. При цьому отримано наступні наукові та практичні результати:

1. Проведено аналіз методичного та алгоритмічного застосування інформаційних транспортних систем та розглянуто сучасні інформаційні транспортні системи виявлення транспортних засобів, визначення та прогнозування інтенсивності транспортного руху,що дозволило обґрунтувати необхідність та актуальність створення інформаційної системи точному визначенні інтенсивності транспортного руху для підвищення якості управління транспортним рухом.

2. Розроблено метод визначення показника завантаженості смуги транспортного руху, який отримав назву TLCR (Traffic Lane Congestion Ratio), для оцінки заторів на одній смузі. Розроблено алгоритм обробки зображень з метою виявлення транспортних засобів на наявній ділянці. Запропоновано метод визначення інтенсивності дорожнього руху за послідовними значеннями показника завантаженості смуги дорожнього руху за даними відеоряду. Правильно визначені значення параметрів завантаженості смуги транспортного руху TLCR та інтенсивності дорожнього руху дозволить системі управління знайти оптимізовані параметри та уникнути заторів.

Експериментально доведено перевагу використання нейронної мережі U-net для задачі визначення інтенсивності руху та показника завантаженості TLCR.

3. Розроблена технологія визначення інтенсивності дорожнього руху за даними відеоряду, що надходять з відеокамери спостереження. Було вдосконалено алгоритм визначення показника завантаженості транспортної ділянки TLCR надає можливість враховувати тільки автомобілі, які рухаються по досліджуваній смузі. Розроблений метод визначення інтенсивності

дорожнього руху на основі послідовних значень показника завантаженості має наступні переваги над іншими подібними системами: швидкість обробки даних, точність, відсутність необхідності додаткового обладнання (наприклад датчики) та низька вартість.

4. Розроблено програмний компонент для визначення інтенсивності дорожнього руху в двох реалізаціях. Експериментально доведено значну перевагу реалізації на основі модуля bioinspired над реалізацією на основі порівняння двох кадрів. Програмний компонент визначення інтенсивності транспортного руху, який розроблений, є складовою частиною інформаційної системи управління транспортним рухом.

Розроблено алгоритм та інформаційну систему для довгострокового прогнозування показника завантаженості транспортної ділянки TLCR для подальшої оцінки стану дорожнього руху. Інформаційна система прогнозування базується на моделі навчання з рекурентною нейронною мережею LSTM. Розроблена система навчається на тренувальному відео отриманих з камер дорожнього руху записного протягом одного тижня для прогнозування показника завантаженості транспортної ділянки TLCR для кожного дня тижня.

5. Розроблено алгоритм оцінювання стану дорожнього руху за показником завантаженості транспортної ділянки TLCR отриманих з зображень отриманих з відеокамер, встановлених у різних місцях міста. Алгоритм дозволяє класифікувати затори за трьома рівнями завантаженості: низька, середня та висока завантаженість. Розроблений алгоритм досліджували на експериментальних даних записів з дорожніх відеокамер та було отримано задовільні результати виявлення та класифікації рівнів завантаженості.

Послідовність обробки та перетворень даних складають нову технологію визначення інтенсивності дорожнього руху, що забезпечує високу точність оцінки інтенсивності руху транспортних засобів на ділянці дорожнього руху. Розроблена технологія може працювати в умовах нестаціонарності параметрів транспортного руху. Завдяки використанню сегментації, замість класифікації, а

також специфічного набору даних для навчання, технологія позбавлена таких недоліків як неправильно підібраний ракурс та відсутність транспортного засобу в існуючих базах даних для навчання. Запропонована система успішно підрахувала транспортні засоби з високою точністю, – середні значення F-міра та точність (Accuracy) досягли 0,9967 та 0,9935 відповідно.

Досліджено точність розробленої інформаційної системи довгострокового прогнозування показника завантаженості транспортної ділянки TLCR, яка базується на моделі навчання з рекурентною нейронною мережею LSTM. Отримана експериментальна середня точність 0,914 для п’яти днів (два вихідних та три робочих дні) демонструє високу ефективність результатів прогнозування.

Одержані результати дозволяють оцінювати стан інтенсивності руху, що дозволило реалізувати інформаційну технологію аналізу транспортних систем в умовах нестаціонарності параметрів транспортного потоку.

ЛІТЕРАТУРА

1. Dahl, M. and Javadi, S. (2020). "Analytical Modeling for a Video-Based Vehicle Speed Measurement Framework". *Sensors*, Vol. 20, no. 1, p. 10.

2. Ki, Y.K. and Baik, D.K. (2006). "Model for accurate speed measurement using double-loop detectors". *IEEE Transactions on Vehicular Technology*, Vol. 55, no. 4, pp.1094-1101.

3. Mei, T.X. and Li, H. (2010). "Measurement of absolute vehicle speed with a simplified inverse model". *IEEE transactions on vehicular technology*, Vol. 59, no. 3, pp.1164-71.

4. Lhomme-Desages, D., Grand, C., Amar, F.B. and Guinot, J.C. (2009). "Doppler-based ground speed sensor fusion and slip control for a wheeled rover". *IEEE/ASME Transactions on mechatronics*, Vol. 14, no. 4, pp.484-92.

5. Ki, Y.K. (2011). "Speed-measurement model utilising embedded triple-loop sensors". *IET Intelligent Transport Systems*, Vol. 5, no. 1, pp.32-37.

6. Buch, N., Velastin, S.A. and Orwell, J. (2011). "A review of computer vision techniques for the analysis of urban traffic". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 12, no. 3, pp.920-39.

7. Nguyen, T.T., Dai Pham, X., Song, J.H., Jin, S., Kim, D. and Jeon, J.W. (2010). "Compensating background for noise due to camera vibration in uncalibrated-camera-based vehicle speed measurement system". *IEEE Transactions on Vehicular Technology*, Vol. 60, no. 1, pp.30-43.

8. Weng, M., Huang, G. and Da, X., 2010, October. A new interframe difference algorithm for moving target detection. In *2010 3rd international congress on image and signal processing* (Vol. 1, pp. 285-289). IEEE.

9. Lan, J., Li, J., Hu, G., Ran, B. and Wang, L. (2014). "Vehicle speed measurement based on gray constraint optical flow algorithm". *Optik*, Vol. 125, no. 1, pp.289-95.

10. Zhao, P. (2011). "Parallel precise speed measurement for multiple moving objects". *Optik*, Vol. 122, no. 22, pp. 2011-15.

11. Doğan, S., Temiz, M.S. and Külür, S. (2010). "Real time speed estimation of moving vehicles from side view images from an uncalibrated video camera". *Sensors*, Vol. 10, no. 5, pp.4805-24.

12. Derrouz, H., Elbouziady, A., Abdelali, H.A., Thami, R.O.H., El Fkihi, S. and Bourzeix, F. (2019). "Moroccan Video Intelligent Transport System: Vehicle Type Classification Based on Three-Dimensional and Two-Dimensional Features". *Ieee Access*, Vol. 7, pp. 72528-37.

13. Pustokhina, I.V., Pustokhin, D.A., Rodrigues, J., Gupta, D., Khanna, A., Shankar, K. et al. (2020). "Automatic Vehicle License Plate Recognition Using Optimal K-Means With Convolutional Neural Network for Intelligent Transportation Systems". *Ieee Access*, Vol. 8, pp. 92907-17.

14. Yamaguchi, K., Nagaya, Y., Ueda, K., Nemoto, H. and Nakagawa, M., 1999, October. A method for identifying specific vehicles using template matching. In *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems* (Cat. No. 99TH8383) (pp. 8-13). IEEE.

15. Mao, Q.C., Sun, H.M., Zuo, L.Q. and Jia, R.S. (2020). "Finding every car: a traffic surveillance multi-scale vehicle object detection method". *Applied Intelligence*, p. 12.

16. Swiderska-Chadaj, Z., Pinckaers, H., van Rijthoven, M., Balkenhol, M., Melnikova, M., Geessink, O., Manson, Q., Litjens, G., van der Laak, J. and Ciompi, F., 2018. Convolutional neural networks for lymphocyte detection in immunohistochemically stained whole-slide images.

17. Wang, Y., Fu, F.F., Lai, F.C., Xu, W.Z., Shi, J.J. and Wang, J.X. (2019). "Haze removal algorithm based on single-images with chromatic properties". *Signal Processing-Image Communication*, Vol. 72, pp. 80-91.

18. Yu, Q.Y., Luo, Y.L., Chen, C.M. and Zheng, X.Y. (2019). "Road Congestion Detection Based on Trajectory Stay-Place Clustering". *Isprs International Journal of Geo-Information*, Vol. 8, no. 6, p. 20.

19. Makhlouf, Y. and Daamouche, A. (2019). "Automatic generation of adaptive structuring elements for road identification in VHR images". *Expert Systems with Applications*, Vol. 119, pp. 342-49.

20. Zedadra, O., Guerrieri, A., Jouandeau, N., Spezzano, G., Seridi, H. and Fortino, G. (2019). Swarm Intelligence and IoT-Based Smart Cities: A Review. In: Cicirelli, F., Guerrieri, A., Mastroianni, C., Spezzano, G. and Vinci, A. (eds.) *Internet of Things for Smart Urban Ecosystems*. Cham: Springer International Publishing Ag.

21. Sengar, S.S. and Mukhopadhyay, S. (2020). "Motion segmentation-based surveillance video compression using adaptive particle swarm optimization". *Neural Computing & Applications*, Vol. 32, no. 15, pp. 11443-57.

22. Li, C.B. and Yang, B. (2019). "CFGVF: An improved correlation filters based visual tracking algorithm". *Optik*, Vol. 192, p. 11.

23. Jian, L.H., Li, Z., Yang, X.M., Wu, W., Ahmad, A. and Jeon, G. (2019). "Combining Unmanned Aerial Vehicles With Artificial-Intelligence Technology for Traffic-Congestion Recognition". *Ieee Consumer Electronics Magazine*, Vol. 8, no. 3, pp. 81-86.

24. Li, J., Chen, S., Zhang, F.B., Li, E.K., Yang, T. and Lu, Z.Y. (2019). "An Adaptive Framework for Multi-Vehicle Ground Speed Estimation in Airborne Videos". *Remote Sensing*, Vol. 11, no. 10, p. 28.

25. Han, S., Yoo, J. and Kwon, S. (2019). "Real-Time Vehicle-Detection Method in Bird-View Unmanned-Aerial-Vehicle Imagery". *Sensors*, Vol. 19, no. 18, p. 17.

26. Fadhil, A.F., Kanneganti, R., Gupta, L., Eberle, H. and Vaidyanathan, R. (2019). "Fusion of Enhanced and Synthetic Vision System Images for Runway and Horizon Detection". *Sensors*, Vol. 19, no. 17, p. 17.

27. Kukkala, V.K., Tunnell, J., Pasricha, S. and Bradley, T. (2018). "Advanced driver-assistance systems: A path toward autonomous vehicles". *IEEE Consumer Electronics Magazine*, Vol. 7, no. 5, pp.18-25.

28. Khan, S.M., Dey, K.C. and Chowdhury, M. (2017). "Real-time traffic state estimation with connected vehicles". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 18, no. 7, pp.1687-99.

29. Eckelmann, S., Trautmann, T., Ußler, H., Reichelt, B. and Michler, O., (2017). "V2v-communication, LIDAR system and positioning sensors for future fusion algorithms in connected vehicles". *Transportation Research Procedia*, Vol. 27, pp.69-76.

30. Van Brummelen, J., O'Brien, M., Gruyer, D. and Najjaran, H. (2018). "Autonomous vehicle perception: The technology of today and tomorrow". *Transportation research part C: emerging technologies*, Vol. 89, pp.384-06.

31. Kim, J.B. (2013)."Development of a robust traffic surveillance system using wavelet support vector machines and wavelet invariant moments". *Inf. Int. Interdiscip. J.*, Vol. 16, pp.3787–3800.

32. Kim, J.B. (2013). "Detection of traffic signs based on eigen-color model and saliency model in driver assistance systems". *International Journal of Automotive Technology*, Vol. 14, no. 3, pp.429-39.

33. Gargoum, S.A., Karsten, L., El-Basyouny, K. and Koch, J.C. (2018). "Automated assessment of vertical clearance on highways scanned using mobile LiDAR technology". *Automation in Construction*, Vol. 95, pp.260-74.

34. Kang, C. and Heo, S.W., 2017, January. Intelligent safety information gathering system using a smart blackbox. In *2017 IEEE International Conference on Consumer Electronics (ICCE)* (pp. 229-230). IEEE.

35. Kim, J.H., Kim, S.K., Lee, S.H., Lee, T.M. and Lim, J., 2018, January. Lane recognition algorithm using lane shape and color features for vehicle black box. In *2018 International Conference on Electronics, Information, and Communication (ICEIC)* (pp. 1-2). IEEE.

36. Rekha, S. and Hithaishi, B.S., 2017, March. Car surveillance and driver assistance using blackbox with the help of GSM and GPS technology. In *2017 International Conference on Recent Advances in Electronics and Communication Technology (ICRAECT)* (pp. 297-301). IEEE.

37. Xing, Y., Lv, C., Chen, L., Wang, H., Wang, H., Cao, D., Velenis, E. and Wang, F.Y. (2018). "Advances in vision-based lane detection: algorithms, integration, assessment, and perspectives on ACP-based parallel vision". *IEEE/CAA Journal of Automatica Sinica*, Vol. 5, no.3, pp.645-61.

38. Chen, Y.C., Su, T.F. and Lai, S.H., 2014, November. Integrated vehicle and lane detection with distance estimation. In *Asian Conference on Computer Vision* (pp. 473-485). Springer, Cham.

39. Kim, G. and Cho, J.S., 2012, October. Vision-based vehicle detection and inter-vehicle distance estimation. In *2012 12th International Conference on Control, Automation and Systems* (pp. 625-629). IEEE.

40. Tram, V.T.B. and Yoo, M. (2018). "Vehicle-to-vehicle distance estimation using a low-resolution camera based on visible light communications". *IEEE Access*, Vol. 6, pp.4521-27.

41. Liu, L.C., Fang, C.Y. and Chen, S.W. (2016). "A novel distance estimation method leading a forward collision avoidance assist system for vehicles on highways". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 18, no.4, pp.937-949.

42. Rezaei, M., Terauchi, M. and Klette, R. (2015). "Robust vehicle detection and distance estimation under challenging lighting conditions". *IEEE transactions on intelligent transportation systems*, Vol. 16, no. 5, pp.2723-43.

43. Huang, D.Y., Chen, C.H., Chen, T.Y., Hu, W.C. and Feng, K.W. (2017). "Vehicle detection and inter-vehicle distance estimation using single-lens video camera on urban/suburb roads". *Journal of Visual Communication and Image Representation*, Vol. 46, pp.250-59.

44. Yang, Z. and Pun-Cheng, L.S.C. (2018). "Vehicle detection in intelligent transportation systems and its applications under varying environments: A review". *Image and Vision Computing*, Vol. 69, pp. 143-54.

45. Thiang, A.T.G. and Lim, R., 2001, March. Type of vehicle recognition using template matching method. In *Proceedings of the International Conference on Electrical, Electronics, Communication and Information* (pp. 7-8).

46. Choi, J.H., Lee, K.H., Cha, K.C., Kwon, J.S., Kim, D.W. and Song, H.K., 2006, September. Vehicle tracking using template matching based on feature points. In *2006 IEEE International Conference on Information Reuse & Integration* (pp. 573-577). IEEE.

47. Sharma, K.(2018). "Feature-based efficient vehicle tracking for a traffic surveillance system". *Computers & Electrical Engineering*, Vol. 70, pp.690-701.

48. Oren, M., Papageorgiou, C., Sinha, P., Osuna, E. and Poggio, T., 1997, June. Pedestrian detection using wavelet templates. In *Proceedings of IEEE computer society Conference on computer vision and pattern recognition* (pp. 193-199). IEEE.

49. Daigavane, P.M., Bajaj, P.R. and Daigavane, M.B., 2011, October. Vehicle detection and neural network application for vehicle classification. In *2011 International Conference on Computational Intelligence and Communication Networks* (pp. 758-762). IEEE.

50. Satzoda, R.K. and Trivedi, M.M.(2015). "Multipart vehicle detection using symmetry-derived analysis and active learning". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 17, no. 4, pp.926-37.

51. Kim, J. (2019). "Automatic vehicle license plate extraction using region-based convolutional neural networks and morphological operations". *Symmetry*, Vol. 11, no. 7, p.882.

52. Wei, Y., Tian, Q., Guo, J., Huang, W. and Cao, J. (2019). "Multi-vehicle detection algorithm through combining Harr and HOG features". *Mathematics and Computers in Simulation*, Vol. 155, pp.130-45.

53. Jazayeri, A., Cai, H., Zheng, J.Y. and Tuceryan, M. (2011). "Vehicle detection and tracking in car video based on motion model". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 12, no. 2, pp.583-95.

54. Chen, S.H. and Chen, R.S., 2011, December. Vision-based distance estimation for multiple vehicles using single optical camera. In *2011 Second International Conference on Innovations in Bio-inspired Computing and Applications* (pp. 9-12). IEEE.

55. Bertozz, M., Broggi, A. and Fascioli, A. (1998). "Stereo inverse perspective mapping: theory and applications". *Image and vision computing*, Vol. 16, no. 8, pp.585-90.

56. Kim, J.B. (2019). "Efficient Vehicle Detection and Distance Estimation Based on Aggregated Channel Features and Inverse Perspective Mapping from a Single Camera". *Symmetry-Basel*, Vol. 11, no. 10, p. 20.

57. Dollár, P., Appel, R., Belongie, S. and Perona, P. (2014). "Fast feature pyramids for object detection". *IEEE transactions on pattern analysis and machine intelligence*, Vol. 36, no. 8, pp.1532-45.

58. Yang, B., Yan, J., Lei, Z. and Li, S.Z., 2014, September. Aggregate channel features for multi-view face detection. In *IEEE international joint conference on biometrics* (pp. 1-8). IEEE.

59. Viola, P. and Jones, M., 2001, December. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001* (Vol. 1, pp. I-I). IEEE.

60. Song, G.Y., Lee, K.Y. and Lee, J.W. (2009). "Vehicle detection using edge analysis and AdaBoost algorithm". *Transactions of the Korean Society of Automotive Engineers*, Vol. 17, no.1, pp.1-11.

61. Zhuang, L., Xu, Y. and Ni, B., 2017, November. Pedestrian detection using ACF based fast R-CNN. In *International Forum on Digital TV and Wireless Multimedia Communications* (pp. 172-181). Springer, Singapore.

62. Kim, J.B. (2015). "Detection of direction indicators on road surfaces using Inverse Perspective Mapping and NN". *J. Inf. Process. Korean*, Vol. 4, pp.201-8.

63. Yang, W., Fang, B. and Tang, Y.Y. (2016). "Fast and accurate vanishing point detection and its application in inverse perspective mapping of structured road". *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 48, no. 5, pp.755-66.

64. Jeong, S.H.and Kim, J.K. (2013). "A study on detection and distance estimation of forward vehicle for FCWS". *Proc. IEEK* 2013,Vol. 1, pp.597–600.

65. Lee, H.S., Oh, S., Jo, D. and Kang, B.Y. (2018). "Estimation of driver's danger level when accessing the center console for safe driving". *Sensors*, Vol. 18, no. 10, p.3392.

66. Yin, J.L., Chen, B.H., Lai, K.H.R. and Li, Y. (2017). "Automatic dangerous driving intensity analysis for advanced driver assistance systems from multimodal driving signals". *IEEE Sensors Journal*, Vol. 18, no. 12, pp.4785-94.

67. ДСТУ ISO 8824:2019. Avtomobilni dorogy. Vyznachennya intensyvnosti ruhu ta skladu transportnogo potoku Автомобільні дороги. Визначення інтенсивності руху та складу транспортного потоку [Highways. Determination of traffic intensity and composition of traffic flow]. (Feb 27, 2019).

68. Kastrinaki, V., Zervakis, M. and Kalaitzakis, K. (2003). "A survey of video processing techniques for traffic applications". *Image and vision computing*, Vol. 21, no. 4, pp.359-81.

69. Li, X., She, Y., Luo, D. and Yu, Z. (2013). "A traffic state detection tool for freeway video surveillance system". *Procedia-Social and Behavioral Sciences*, Vol. 96, pp.2453-61.

70. Wei, L. and Hong-ying, D. (2016). "Real-time road congestion detection based on image texture analysis". *Procedia engineering*, Vol. 137, pp.196-201.

71. Qiao, Y. and Shi, Z. (2012). "Traffic parameters detection using edge and texture. *Procedia Engineering*", Vol. 29, pp.3858-62.

72. Song, B., Han, L.Q., Zhong, Y.X. and Wang, X.J. (2011). "All-day traffic states recognition system without vehicle segmentation". *The Journal of China Universities of Posts and Telecommunications*, Vol. 18, pp.1-11.

73. Asmaa, O., Mokhtar, K. and Abdelaziz, O. (2013). Road traffic density estimation using microscopic and macroscopic parameters. *Image and Vision Computing*, Vol. 31, no. 11, pp.887-94.

74. Morales, D.I., Moctezuma, M. and Parmiggiani, F., 2004, September. Urban edge detection by texture analysis. In *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium* (Vol. 6, pp. 3826-3828). IEEE.

75.Haralick, R.M., Shanmugam, K. and Dinstein, I.H. (1973). Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, Vol. 6, pp.610-21.

76. Yao, B., Chen, C., Cao, Q., Jin, L., Zhang, M., Zhu, H. et al. (2017). "Short-Term Traffic Speed Prediction for an Urban Corridor". *Computer-Aided Civil and Infrastructure Engineering*, Vol. 32, no. 2, pp. 154-69.

77. Vlahogianni, E.I., Karlaftis, M.G. and Golias, J.C. (2014). "Short-term traffic forecasting: Where we are and where we're going". *Transportation Research Part C: Emerging Technologies*, Vol. 43, pp.3-19.

78. Chan, K.Y., Dillon, T.S., Singh, J. and Chang, E. (2011). "Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and Levenberg–Marquardt algorithm". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 13, no. 2, pp. 644-54.

79. Sun, S., Zhang, C. and Yu, G. (2006). "A Bayesian network approach to traffic flow forecasting". *IEEE Transactions on intelligent transportation systems*, Vol. 7, no. 1, pp. 124-32.

80. Kumar, S.V. and Vanajakshi, L. (2015). "Short-term traffic flow prediction using seasonal ARIMA model with limited input data". *European Transport Research Review*, Vol. 7, no. 3, p.21.

81. Zhou, T., Jiang, D., Lin, Z., Han, G., Xu, X. and Qin, J. (2019). "Hybrid dual Kalman filtering model for short-term traffic flow forecasting". *Iet Intelligent Transport Systems*, Vol. 13, no. 6, pp. 1023-32.

82. Ojeda, L.L., Kibangou, A.Y. and De Wit, C.C., 2013, June. Adaptive Kalman filtering for multi-step ahead traffic flow prediction. In *2013 American Control Conference* (pp. 4724-4729). IEEE.

83. Bezuglov, A. and Comert, G. (2016). "Short-term freeway traffic parameter prediction: Application of grey system theory models". *Expert Systems with Applications*, Vol. 62, pp. 284-92.

84. Mao, S., Xiao, X., Gao, M., Wang, X. and He, Q. (2018). "Nonlinear Fractional Order Grey Model of Urban Traffic Flow Short-Term Prediction". *Journal of Grey System*, Vol. 30, no. 4.

85. Okutani, I. and Stephanedes, Y.J. (1984). "Dynamic prediction of traffic volume through Kalman filtering theory". *Transportation Research Part B: Methodological*, Vol. 18, no. 1, pp.1-11.

86. Sun, H., Liu, H.X., Xiao, H., He, R.R. and Ran, B. (2003). "Use of local linear regression model for short-term traffic forecasting". *Transportation Research Record*, Vol. 1836, no. 1, pp.143-50.

87. Dougherty, M. (1995). "A review of neural networks applied to transport". *Transportation Research Part C: Emerging Technologies*, Vol. 3 no. 4, pp.247-60.

88. Postorino, M.N. and Sarnè, G.M. (1995). "Mobility forecast in an urban area through the use of neural networks", *Appl. Adv. Technol. Transp. Eng.*, Vol. 13, no. 1, pp. 213-17.

89. More, R., Mugal, A., Rajgure, S., Adhao, R.B. and Pachghare, V.K., 2016, December. Road traffic prediction and congestion control using Artificial Neural Networks. In *2016 International Conference on Computing, Analytics and Security Trends (CAST)* (pp. 52-57). IEEE.

90. Clark, S.(2003). "Traffic prediction using multivariate nonparametric regression". *Journal of transportation engineering*, Vol. 129, no. 2, pp.161-68.

91. Tang, Y.F., Lam, W.H. and Ng, P.L.(2003). "Comparison of four modeling techniques for short-term AADT forecasting in Hong Kong". *Journal of Transportation Engineering*, Vol. 129, no. 3, pp.271-77.

92. Zhang, G., Patuwo, B.E. and Hu, M.Y. (1998). "Forecasting with artificial neural networks: The state of the art". *International journal of forecasting*, Vol. 14, no. 1, pp.35-62.

93. Liang, X.R. and Wei, Y.X., 2007. Freeway traffic flow modeling based on recurrent neural network and wavelet transform. In *International Conference on Transportation Engineering 2007* (pp. 1088-1093).

94. Tan, M.C., Wong, S.C., Xu, J.M., Guan, Z.R. and Zhang, P. (2009). "An aggregation approach to short-term traffic flow prediction". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 10, no. 1, pp.60-69.

95. Lv, Y., Duan, Y., Kang, W., Li, Z. and Wang, F.Y. (2014). "Traffic flow prediction with big data: a deep learning approach". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 16, no. 2, pp.865-73.

96. Yu, D., Liu, Y. and Yu, X., 2016, September. A data grouping CNN algorithm for short-term traffic flow forecasting. In *Asia-Pacific Web Conference* (pp. 92-103). Springer, Cham.

97. Xu, Y., Hu, D.-w. and Su, B. (2017). "Short-term traffic flow prediction based on optimised support vector regression". *International Journal of Applied Decision Sciences*, Vol. 10, no. 4, pp. 305-14.

98. Dai, X., Fu, R., Zhao, E., Zhang, Z., Lin, Y., Wang, F.-Y. et al. (2019). "DeepTrend 2.0: A light-weighted multi-scale traffic prediction model using detrending". *Transportation Research Part C-Emerging Technologies*, Vol. 103, pp. 142-57.

99. Li, L., Qin, L., Qu, X., Zhang, J., Wang, Y. and Ran, B. (2019a). "Day-ahead traffic flow forecasting based on a deep belief network optimized by the multi-objective particle swarm algorithm". *Knowledge-Based Systems*, Vol. 172, pp. 1-14.

100. Zhao, Z., Chen, W., Wu, X., Chen, P.C.Y. and Liu, J. (2017). "LSTM network: a deep learning approach for short-term traffic forecast". *Iet Intelligent Transport Systems*, Vol. 11, no. 2, pp. 68-75.

101.Polson, N.G. and Sokolov, V.O. (2017). "Deep learning for short-term traffic flow prediction". *Transportation Research Part C-Emerging Technologies*, Vol. 79, pp. 1-17.

102.Xiao, X.P. and Duan, H.M. (2020). "A new grey model for traffic flow mechanics". *Engineering Applications of Artificial Intelligence*, Vol. 88, p. 12.

103.Wang, K.C. and Li, Q. (2011). "Pavement smoothness prediction based on fuzzy and gray theories". *Computer-Aided Civil and Infrastructure Engineering*, Vol. 26, no. 1, pp. 69-76.

104.Khashei, M. and Bijari, M. (2011). "A novel hybridization of artificial neural networks and ARIMA models for time series forecasting". *Applied Soft Computing, Vol*. 11, no. 2, pp. 2664-75.

105.Chen, Y., Zhao, Y. and Yan, P., 2016, August. Daily ETC traffic flow time series prediction based on k-NN and BP neural network. In *International Conference of Pioneering Computer Scientists, Engineers and Educators* (pp. 135-146). Springer, Singapore.

106.Zhang, Y. and Liu, Y., 2009. Application of combined forecasting models to intelligent transportation systems. In *Opportunities and Challenges for Next-Generation Applied Intelligence* (pp. 181-186). Springer, Berlin, Heidelberg.

107.Tang, J., Li, L., Hu, Z. and Liu, F. (2019b). "Short-term traffic flow prediction considering spatio-temporal correlation: A hybrid model combing type-2 fuzzy C-means and artificial neural network". *IEEE Access*, Vol. 7, pp. 101009-18.

108.Zhang, Y., Zhang, Y. and Haghani, A. (2014). "A hybrid short-term traffic flow forecasting method based on spectral analysis and statistical volatility model". *Transportation Research Part C-Emerging Technologies*, Vol. 43, pp. 65-78.

109.Feng, W., Chen, H. and Zhang, Z., 2017, November. Short-term traffic flow prediction based on wavelet function and extreme learning machine. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)* (pp. 531-535). IEEE.

110.Wu, Y., Tan, H., Qin, L., Ran, B. and Jiang, Z. (2018). "A hybrid deep learning based traffic flow prediction method and its understanding". *Transportation Research Part C: Emerging Technologies*, Vol. 90, pp.166-80.

111.Zheng, Z., Pan, L. and Pholsena, K., 2018, June. Mode decomposition based hybrid model for traffic flow prediction. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)* (pp. 521-526). IEEE.

112. OpenCV 2.4.13.7 Documentation. Structural Analysis and Shape Descriptors. URL: https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html(last accessed 2020/04/22).

113. Hérault J. Vision: Images, Signals and Neural Networks-Models of Neural Processing in Visual Perception / Hérault J. // World Scientific, 2010.

114. De Lavarène, B.C., Alleysson, D., Durette, B. and Hérault, J., 2007, September. Efficient demosaicing through recursive filtering. In *2007 IEEE International Conference on Image Processing* (Vol. 2, pp. II-189). IEEE.

115. OpenCV [Електронний ресурс]. - Режим доступу: https://docs.opencv.org/3.2.0/index.html.

116. Ярышев С.Н., 2011. *Цифровые методы обработки видеоинформации и видеоаналитика: учебное пособие*. СПб.: СПбГУ ИТМО, 83 с.

117. Stetsenko I.V. and Stelmakh O.P. (2020). "Traffic Lane Congestion Ratio Evaluation by Video Data". *Advances in Intelligent Systems and Computing*, Vol. 1019,pp. 172-81.

118. Ronneberger, O., Fischer, P. and Brox, T., 2015, October. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

119. Алгоритм определения движения через сравнение двух кадров [Електронний ресурс]. URL: https://habr.com/post/134635.

120. Retina : a Bio mimetic human retina model [Електронний ресурс]. - Режим доступу: https://docs.opencv.org/3.0-beta/modules/bioinspired/doc/retina/index.html#retina

121. Kingma, D.P. and Ba, J., 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR 2015)* (pp. 1-13). Kingma.

122. Khizhnyak, A.V. and Kuzmenok, M.D. (2018). "Recognizing images using convolutional networks". *Digital Library of Polotsk State University*, http://elib.psu.by:8080/handle/123456789/22222, last accessed 2019/04/25.

123. Chen, Y. and Hu, W. (2020). "Robust Vehicle Detection and Counting Algorithm Adapted to Complex Traffic Environments with Sudden Illumination Changes and Shadows". *Sensors*, Vol. 20, no. 9, p. 21.

124. Bouvié, C., Scharcanski, J., Barcellos, P. and Escouto, F.L., 2013. Tracking and counting vehicles in traffic video sequences using particle filtering. In *Proceedings of the 2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)* (pp. 812-815). IEEE.

125. Quesada, J. and Rodriguez, P., 2016. Automatic vehicle counting method based on principal component pursuit background modeling. In *Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP)* (pp. 3822-3826). IEEE.

126. Yang, H. and Qu, S. (2018). "Real-time vehicle detection and counting in complex traffic scenes using background subtraction model with low-rank decomposition". *IET Intelligent Transport Systems*, Vol. 12, no. 1, pp. 75–85.

127. Abdelwahab, M.A. (2019). "Fast approach for efficient vehicle counting". *Electronics Letters*, Vol. 55, no. 1, pp. 20–21.

128. Wang, H.W., Peng, Z.R., Wang, D., Meng, Y., Wu, T., Sun, W. and Lu, Q.C. (2020). "Evaluation and prediction of transportation resilience under extreme weather events: A diffusion graph convolutional approach". *Transportation research part C: emerging technologies*.Vol. 115, pp. 102-19.

129. Zahid, M., Chen, Y., Jamal, A. and Memon, M.Q. (2020). "Short term traffic state prediction via hyperparameter optimization based classifiers". *Sensors*, Vol. 20, no. 3, pp. 685-707.

ДОДАТОК А

Частина програмного коду розробленої інформаційної системи

ЛістингА.1. – МодульLineModel.py

```
from PySide2.QtSql import *
from PySide2 import QtCore
from PySide2.QtCore import Qt


class LineModel(QSqlQueryModel):
    _sql = ""
    IdRole = Qt.UserRole + 1
    CameraRole = Qt.UserRole + 2
    CoordXRole = Qt.UserRole + 3
    Coord2XRole = Qt.UserRole + 4
    Coord3XRole = Qt.UserRole + 5
    Coord4XRole = Qt.UserRole + 6
    CoordYRole = Qt.UserRole + 7
    Coord2YRole = Qt.UserRole + 8
    Coord3YRole = Qt.UserRole + 9
    Coord4YRole = Qt.UserRole + 10
    CoordsRole = Qt.UserRole + 11

    def __init__(self, sql='', *args, **kwargs):
        super(LineModel, self).__init__(*args, **kwargs)
        self.__sql = sql

    def flags(self, index: QtCore.QModelIndex) -> QtCore.Qt.ItemFlags:
        return Qt.ItemIsEnabled | Qt.ItemIsSelectable

    def data(self, index, role):
        if role == Qt.DisplayRole:
            return '({}x{}),({}x{}), ({}x{}), ({}x{})'.format(self.data(index, LineModel.CoordXRole),
                                                self.data(index, LineModel.CoordYRole),
                                                self.data(index, LineModel.Coord2XRole),
                                                self.data(index, LineModel.Coord2YRole),
                                                self.data(index, LineModel.Coord3XRole),
                                                self.data(index, LineModel.Coord3YRole),
                                                self.data(index, LineModel.Coord4XRole),
                                                self.data(index, LineModel.Coord4YRole))
```

```
            # return self.record(index.row()).value(LineModel.TextRole - Qt.UserRole - 1)
        if role == LineModel.CoordsRole:
            return [[self.data(index, LineModel.CoordXRole), self.data(index, LineModel.CoordYRole)],
                    [self.data(index, LineModel.Coord2XRole), self.data(index, LineModel.Coord2YRole)],
                    [self.data(index, LineModel.Coord3XRole), self.data(index, LineModel.Coord3YRole)],
                    [self.data(index, LineModel.Coord4XRole), self.data(index, LineModel.Coord4YRole)]]

        if role > Qt.UserRole:
            return self.record(index.row()).value(role - Qt.UserRole - 1)
        return super().data(index, role)

    def refresh(self):
        self.setQuery(self.__sql)
        self.layoutChanged.emit()

    def open(self):
        self.refresh()
```

ЛістингА.2. – Модуль SingleCameraModel.py

```
from models.Observed import Observed, event
from PySide2 import QtCore
from models.camera.CameraModel import CameraModel
from models.line.SingleLineModel import SingleLineModel
from models.line.LineSql import LineSql
from models.line.LineModel import LineModel
from classes.Video import Video
from classes.CVThread import CVThread
from classes.LoadVideoThread import LoadVideoThread
import os


class SingleCameraModel(Observed, QtCore.QObject):
    def __init__(self, index):
        Observed.__init__(self)
        QtCore.QObject.__init__(self)
        self.__status = ''
        self.__progress_max = 100
        self.__progress_value = 0
        self.__index = index
        self.__busy = False
        self.pixmap = index.data(QtCore.Qt.DecorationRole)
```

```
    self.title = index.data(CameraModel.TextRole)
    self.gplmod = self.__index.data(CameraModel.GplModRole)
    self.server = self.__index.data(CameraModel.ServerRole)
    self.id = self.__index.data(CameraModel.IdRole)
    self.lines = []
    self.__init_lines()

def __init_lines(self):
    self.__line_model = LineModel(LineSql.all_from_camera(self.id))
    self.__line_model.open()
    for i in range(0, self.__line_model.rowCount()):
        self.lines.append(SingleLineModel(self.__line_model.index(i, 0)))

def __set_busy(self, value):
    self.__busy = value

def get_status(self):
    return self.__status

@event
def set_status(self, value):
    self.__status = value

def get_progress_max(self):
    return self.__progress_max

@event
def set_progress_max(self, value):
    self.__progress_max = value

def get_progress_value(self):
    return self.__progress_value

@event
def set_progress_value(self, value):
    self.__progress_value = value

def load_video(self):
    if self.__busy:
        print('busy')
        return
    self.__set_busy(True)
```

```
        self.status = 'Завантаженнявідео'

        thread = LoadVideoThread(self.id, self.gplmod, self.server)
        thread.signals.updateProgress.connect(self.set_progress_value)
        thread.signals.updateProgressMax.connect(self.set_progress_max)
        thread.signals.updateStatus.connect(self.set_status)
        thread.signals.finished.connect(lambda: self.__set_busy(False))
        thread.start()

    def run_cv(self):
        if self.__busy:
            print('busy')
            return
        self.__set_busy(True)
        self.status = 'Обробказображень'

        thread = CVThread(self.id)
        thread.signals.updateProgress.connect(self.set_progress_value)
        thread.signals.updateProgressMax.connect(self.set_progress_max)
        thread.signals.updateStatus.connect(self.set_status)
        thread.signals.finished.connect(lambda: self.__set_busy(False))
        thread.start()

    status = property(get_status, set_status)
    progress_max = property(get_progress_max, set_progress_max)
    progress_value = property(get_progress_value, set_progress_value)
```

ЛістингА.3. – Модуль CameraModel.py

```
from models.camera.BaseCameraModel import BaseCameraModel
from PySide2.QtCore import Qt
from PySide2.QtGui import QPixmap
from PySide2.QtGui import QPixmapCache
import requests


class CameraModel(BaseCameraModel):
    def __init__(self, sql='', *args, **kwargs):
        super(CameraModel, self).__init__(*args, **kwargs)
        self._sql = sql

    def data(self, index, role):
```

```
        if role == Qt.DecorationRole:
            image_url = self.record(index.row()).value(CameraModel.ImageRole - Qt.UserRole - 1)
            pm = QPixmap()
            if not QPixmapCache.find(image_url, pm):
                pm.loadFromData(requests.get(image_url).content)
                pm = pm.scaledToHeight(50)
                QPixmapCache.insert(image_url, pm)
            return pm
        return super().data(index, role)

    def delete(self, index):
        id = index.data(CameraModel.IdRole)
        if id is not None:
            # Database.delete_selected_camera(id)
            self.refresh()
```

ЛістингА.4. – МодульSignleLineModel.py

```
from models.Observed import Observed, event
from PySide2 import QtCore
from models.line.LineModel import LineModel
from classes.ImageDataSetThread import ImageDataSetThread
from classes.LearnImageNetThread import LearnImageNetThread
from classes.RunImageNetThread import RunImageNetThread


class SingleLineModel(Observed):
    def __init__(self, index):
        super().__init__()
        self.__status = ''
        self.__progress_max = 100
        self.__progress_value = 0
        self.__index = index
        self.__busy = False
        self.title = index.data(QtCore.Qt.DisplayRole)
        self.camera_id = index.data(LineModel.CameraRole)
        self.id = index.data(LineModel.IdRole)
        self.coords = index.data(LineModel.CoordsRole)

    def __set_busy(self, value):
        self.__busy = value
```

```
def get_status(self):
    return self.__status

@event
def set_status(self, value):
    self.__status = value

def get_progress_max(self):
    return self.__progress_max

@event
def set_progress_max(self, value):
    self.__progress_max = value

def get_progress_value(self):
    return self.__progress_value

@event
def set_progress_value(self, value):
    self.__progress_value = value

def generate_dataset(self):
    if self.__busy:
        print('busy')
        return
    self.__set_busy(True)
    self.status = 'Генераціядатасету'

    thread = ImageDataSetThread(self.camera_id, self.id, self.coords)
    thread.signals.updateProgress.connect(self.set_progress_value)
    thread.signals.updateProgressMax.connect(self.set_progress_max)
    thread.signals.updateStatus.connect(self.set_status)
    thread.signals.finished.connect(lambda: self.__set_busy(False))
    thread.start()

def train(self):
    if self.__busy:
        print('busy')
        return
    self.__set_busy(True)
    self.status = 'Навчаннямережі'
```

```
        thread = LearnImageNetThread(self.camera_id, self.id)
        thread.signals.updateProgress.connect(self.set_progress_value)
        thread.signals.updateProgressMax.connect(self.set_progress_max)
        thread.signals.updateStatus.connect(self.set_status)
        thread.signals.finished.connect(lambda: self.__set_busy(False))
        thread.start()
        # thread = RunImageNetThread('D:/intensity/18/cam18stream_1579869707.mp4', self.coords)
        # thread.start()

    status = property(get_status, set_status)
    progress_max = property(get_progress_max, set_progress_max)
    progress_value = property(get_progress_value, set_progress_value)
```

ЛістингА.5. – МодульCalculateParams.py

```
from threading import Thread, Event
import time
import numpy as np
import os
import cv2
from skimage.util import img_as_float
from classes.Video import Video
from classes.Line import Line
from classes.ImageNet import ImageNet
from models.line.LineSql import LineSql
from models.line.LineModel import LineModel



class LineThread(Thread):
    def __init__(self, index, images, event_start, event_end):
        super().__init__()
        self.__index = index
        self.__images = images
        self.__event_start = event_start
        self.__event_end = event_end
        self.__line_id = self.__index.data(LineModel.IdRole)
        self.__coords = self.__index.data(LineModel.CoordsRole)
        self.__line = Line(self.__coords)
        self.__image_net = None

    def __load_image_net(self):
```

```
    if self.__image_net:
        return
    self.__image_net = ImageNet()
    weights_dir = os.path.join(os.getcwd(), 'weights')
    weights_file_name = os.path.join(weights_dir, 'image_net_{}.weights'.format(self.__line_id))
    print(weights_file_name, self.__line_id)
    if os.path.isfile(weights_file_name) or os.path.isfile('{}.index'.format(weights_file_name)):
        self.__image_net.load_weights(weights_file_name)
        print('loaded', self.__line_id)

def run(self):
    while True:
        self.__event_start.wait()
        self.__event_start.clear()

        # generate x data for image net
        print('get train_x data from images', self.__line_id)
        size = len(self.__images)
        train_x = np.zeros((size, ImageNet.size_img, ImageNet.size_img, 3), dtype='float32')
        for i in range(0, size):
            train_x[i] = np.array(img_as_float(self.__line.get_image(self.__images[i])))

        if self.__line_id != 6:
            continue
        # predictions
        print('image net prediction', self.__line_id)
        self.__load_image_net()
        predictions = self.__image_net.predict(train_x)

        continue
        # calculate tlcr, intensity
        print('calculate tlcr, intensity from prediction', self.__line_id)
        size_predictions = len(predictions)
        tlcr = 0
        intensity = 0
        for i in range(0, size_predictions):
            tlcr += self.__line.get_tlcr(predictions[i])
        tlcr = tlcr / size_predictions
        print(tlcr, intensity, self.__line_id)

        self.__event_end.set()
```

```
class CameraThread(Thread):
    def __init__(self, camera_id):
        super().__init__()
        self.__camera_id = camera_id
        self.__images = []
        self.__images_ready_event = Event()

        self.__line_threads = []
        self.__line_events = []
        self.__line_model = LineModel(LineSql.all_from_camera(self.__camera_id))
        self.__line_model.open()
        for i in range(0, self.__line_model.rowCount()):
            self.__line_events.append(Event())
            self.__line_threads.append(
                LineThread(self.__line_model.index(i, 0), self.__images, self.__images_ready_event,
                        self.__line_events[i]))

    def run(self):
        for thread in self.__line_threads:
            thread.setDaemon(True)
            thread.start()
        while True:
            start_time = time.time()
            self.__images.clear()
            self.__images_ready_event.set()

            for event in self.__line_events:
                event.wait()
                event.clear()
            print('after all threads')
            print("all %s seconds" % (time.time() - start_time))
            time.sleep(50)


class CalculateParams:
    def __init__(self):
        pass

    def run(self):
        camera_thread = CameraThread(18)
        camera_thread.setDaemon(True)
```

```
        camera_thread.start()
```

ЛістингА.6. – МодульCVThread.py

```
from threading import Thread
import time
import os
import subprocess
import re
from classes.Video import Video
from PySide2 import QtCore


class CVThreadSignals(QtCore.QObject):
    updateProgress = QtCore.Signal(int)
    updateStatus = QtCore.Signal(str)
    updateProgressMax = QtCore.Signal(int)
    finished = QtCore.Signal()


class CVThread(Thread):
    __max_files = 200

    def __init__(self, camera_id):
        super().__init__()
        self.__camera_id = camera_id
        self.signals = CVThreadSignals()

    def run(self):
        camera_dir = os.path.join(Video.video_folder, str(self.__camera_id))
        orig_dir = os.path.join(camera_dir, Video.orig_folder)
        res_dir = os.path.join(camera_dir, Video.res_folder)
        if not os.path.exists(orig_dir):
            os.makedirs(orig_dir)
        if not os.path.exists(res_dir):
            os.makedirs(res_dir)
        list_files = [i for i in os.listdir(camera_dir) if i.endswith('.mp4')]
        if not list_files:
            self.signals.updateStatus.emit('немаєвідеофайлів')
            self.signals.finished.emit()
            return
        # list_files = list_files[0:200]
```

```
            self.signals.updateProgressMax.emit(len(list_files) + 1)
            pattern = re.compile(r'[^\s]+.mp4')

            len_files = len(list_files)
            print(len_files)
            for i in range(0, len_files, self.__max_files):
                start = i
                end = i + self.__max_files if len_files > i + self.__max_files else len_files

                app = os.path.join(Video.video_folder, 'PrepareData.exe')
                args = [app, '-u', ','.join(list_files[start:end]), '-d', camera_dir]
                process = subprocess.Popen(args, stdin=subprocess.PIPE, stdout=subprocess.PIPE, shell=True,
universal_newlines=True)
                self.signals.updateProgress.emit(1)
                for line in process.stdout:
                    print(line)
                    result = pattern.search(line)
                    if result:
                        file = result.group(0)
                        if file in list_files:
                            number = list_files.index(file)
                            self.signals.updateProgress.emit(number + 2)
                            self.signals.updateStatus.emit('Обробкафайлу: {}\n'.format(file))
                return_code = process.wait()
            self.signals.finished.emit()
```

ImageDataSetThread.py

```
from threading import Thread
import time
import os
import subprocess
import re
import cv2
from classes.Video import Video
from classes.Line import Line
from PySide2 import QtCore


class ImageDataSetThreadSignals(QtCore.QObject):
    updateProgress = QtCore.Signal(int)
```

```
    updateStatus = QtCore.Signal(str)
    updateProgressMax = QtCore.Signal(int)
    finished = QtCore.Signal()


class ImageDataSetThread(Thread):
    __minimum_tlcr = 60

    def __init__(self, camera_id, id, coords):
        super().__init__()
        self.__camera_id = camera_id
        self.__id = id
        self.__coords = coords
        self.signals = ImageDataSetThreadSignals()

    def run(self):
        # dirs
        camera_dir = os.path.join(Video.video_folder, str(self.__camera_id))
        orig_dir = os.path.join(camera_dir, Video.orig_folder)
        res_dir = os.path.join(camera_dir, Video.res_folder)
        ds_dir = os.path.join(camera_dir, str(self.__id))
        orig_ds_dir = os.path.join(ds_dir, Video.orig_folder)
        res_ds_dir = os.path.join(ds_dir, Video.res_folder)
        if not os.path.exists(orig_ds_dir):
            os.makedirs(orig_ds_dir)
        if not os.path.exists(res_ds_dir):
            os.makedirs(res_ds_dir)

        line = Line(self.__coords)
        # files
        list_files = os.listdir(res_dir)
        self.signals.updateProgressMax.emit(len(list_files))
        i = 1
        for file in list_files:
            orig_image = cv2.imread(os.path.join(orig_dir, file))
            res_image = cv2.imread(os.path.join(res_dir, file), 0)
            final_orig, final_res = line.get_dataset_images(orig_image, res_image)
            tlcr = line.get_tlcr(final_res)
            if tlcr >= self.__minimum_tlcr:
                cv2.imwrite(os.path.join(orig_ds_dir, file), final_orig)
                cv2.imwrite(os.path.join(res_ds_dir, file), final_res)
            self.signals.updateProgress.emit(i)
```

```
            i += 1
        self.signals.finished.emit()
```

ЛістингА.7. – МодульImageNet.py

```
from __future__ import absolute_import, division, print_function, unicode_literals
import tensorflow as tf
from keras.backend.tensorflow_backend import set_session
from pympler import muppy, summary

import numpy as np
import os
import time

from tensorflow.keras import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import binary_crossentropy
from tensorflow.keras.layers import Input, Conv2D, Conv2DTranspose, MaxPooling2D, concatenate, Dropout
from tensorflow.keras import backend as K
from keras.utils.generic_utils import get_custom_objects


class ImageNet:
    size_img = 256

    def __init__(self):
        #                                                   config                                                  =
tf.compat.v1.ConfigProto(gpu_options=tf.GPUOptions(per_process_gpu_memory_fraction=0.7),
        #                             allow_soft_placement=True)
        # config.gpu_options.allow_growth = True
        # # config.gpu.set_per_process_memory_fraction(0.4)
        # config.gpu_options.per_process_gpu_memory_fraction = 0.9
        # set_session(tf.compat.v1.Session(config=config))

        #                                                   config                                                  =
tf.compat.v1.ConfigProto(gpu_options=tf.GPUOptions(per_process_gpu_memory_fraction=0.7),
        #                             allow_soft_placement=True)
        # config.gpu_options.allow_growth = True
        # # config.gpu.set_per_process_memory_fraction(0.4)
        # set_session(tf.compat.v1.Session(config=config))
        os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
        self.init()
```

```
def build_model(self, input_layer, start_neurons):
    conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(input_layer)
    conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(conv1)
    pool1 = MaxPooling2D((2, 2))(conv1)
    pool1 = Dropout(0.25)(pool1)

    conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(pool1)
    conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(conv2)
    pool2 = MaxPooling2D((2, 2))(conv2)
    pool2 = Dropout(0.5)(pool2)

    conv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(pool2)
    conv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(conv3)
    pool3 = MaxPooling2D((2, 2))(conv3)
    pool3 = Dropout(0.5)(pool3)

    conv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(pool3)
    conv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(conv4)
    pool4 = MaxPooling2D((2, 2))(conv4)
    pool4 = Dropout(0.5)(pool4)

    # Middle
    convm = Conv2D(start_neurons * 16, (3, 3), activation="relu", padding="same")(pool4)
    convm = Conv2D(start_neurons * 16, (3, 3), activation="relu", padding="same")(convm)

    deconv4 = Conv2DTranspose(start_neurons * 8, (3, 3), strides=(2, 2), padding="same")(convm)
    uconv4 = concatenate([deconv4, conv4])
    uconv4 = Dropout(0.5)(uconv4)
    uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(uconv4)
    uconv4 = Conv2D(start_neurons * 8, (3, 3), activation="relu", padding="same")(uconv4)

    deconv3 = Conv2DTranspose(start_neurons * 4, (3, 3), strides=(2, 2), padding="same")(uconv4)
    uconv3 = concatenate([deconv3, conv3])
    uconv3 = Dropout(0.5)(uconv3)
    uconv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(uconv3)
    uconv3 = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same")(uconv3)

    deconv2 = Conv2DTranspose(start_neurons * 2, (3, 3), strides=(2, 2), padding="same")(uconv3)
    uconv2 = concatenate([deconv2, conv2])
    uconv2 = Dropout(0.5)(uconv2)
    uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(uconv2)
```

```
    uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same")(uconv2)

    deconv1 = Conv2DTranspose(start_neurons * 1, (3, 3), strides=(2, 2), padding="same")(uconv2)
    uconv1 = concatenate([deconv1, conv1])
    uconv1 = Dropout(0.5)(uconv1)
    uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(uconv1)
    uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same")(uconv1)

    uncov1 = Dropout(0.5)(uconv1)
    output_layer = Conv2D(1, (1, 1), padding="same", activation="sigmoid")(uconv1)
    return output_layer

def dice_coeficient(self, y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred = K.cast(y_pred, 'float32')
    y_pred_f = K.cast(K.greater(K.flatten(y_pred), 0.5), 'float32')
    intersection = y_true_f * y_pred_f
    score = 2. * K.sum(intersection) / (K.sum(y_true_f) + K.sum(y_pred_f))
    return score

def dice_loss(self, y_true, y_pred):
    smooth = 1.
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = y_true_f * y_pred_f
    score = (2. * K.sum(intersection) + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)
    return 1. - score

def bce_dice_loss(self, y_true, y_pred):
    return binary_crossentropy(y_true, y_pred) + self.dice_loss(y_true, y_pred)

def get_iou_vector(self, A, B):
    # Numpy version

    batch_size = A.shape[0]
    metric = 0.0
    for batch in range(batch_size):
        t, p = A[batch], B[batch]
        true = np.sum(t)
        pred = np.sum(p)

        # deal with empty mask first
```

```
        if true == 0:
            metric += (pred == 0)
            continue

        intersection = np.sum(t * p)
        union = true + pred - intersection
        iou = intersection / union

        iou = np.floor(max(0, (iou - 0.45) * 20)) / 10

        metric += iou

    metric /= batch_size
    return metric

def my_iou_metric(self, label, pred):
    return tf.py_func(self.get_iou_vector, [label, pred > 0.5], tf.float64)

def init(self):
    get_custom_objects().update({'bce_dice_loss': self.bce_dice_loss})
    get_custom_objects().update({'dice_loss': self.dice_loss})
    get_custom_objects().update({'dice_coef': self.dice_coeficient})
    get_custom_objects().update({'my_iou_metric': self.my_iou_metric})

    input_layer = Input((self.size_img, self.size_img, 3))
    output_layer = self.build_model(input_layer, 16)
    self.model = Model(input_layer, output_layer)
    self.model.compile(loss=self.bce_dice_loss, optimizer=Adam(lr=1e-3), metrics=[self.my_iou_metric])
    # self.model.save_weights('./keras.weights')
    # self.model.load_weights('./kerasFINAL.weights', by_name=False)

def train(self, train_x, train_y, epochs=10):
    history = self.model.fit(train_x, train_y, batch_size=32, epochs=epochs, verbose=1, validation_split=0.1)
    # self.model.save_weights('./kerasFINAL.weights')

def save_weights(self, file_name):
    self.model.save_weights(file_name)

def load_weights(self, file_name):
    self.model.load_weights(file_name, by_name=False)

def predict(self, image_array):
```

```
        return self.model.predict(image_array)
```

ЛістингА.8. – МодульLearnImageNetThread.py

```
from threading import Thread
import time
import os
import cv2
import numpy as np
from skimage.util import img_as_float
from classes.Video import Video
from classes.ImageNet import ImageNet
from PySide2 import QtCore


class LearnImageNetThreadSignals(QtCore.QObject):
    updateProgress = QtCore.Signal(int)
    updateStatus = QtCore.Signal(str)
    updateProgressMax = QtCore.Signal(int)
    finished = QtCore.Signal()


class LearnImageNetThread(Thread):
    __max_files = 500
    __epochs = 5

    def __init__(self, camera_id, id):
        super().__init__()
        self.__camera_id = camera_id
        self.__id = id
        self.signals = LearnImageNetThreadSignals()

    def run(self):
        # dirs
        camera_dir = os.path.join(Video.video_folder, str(self.__camera_id))
        ds_dir = os.path.join(camera_dir, str(self.__id))
        orig_ds_dir = os.path.join(ds_dir, Video.orig_folder)
        res_ds_dir = os.path.join(ds_dir, Video.res_folder)

        # files
        list_files = os.listdir(res_ds_dir)
        len_files = len(list_files)
        self.signals.updateProgressMax.emit(len_files + len_files * self.__epochs)
```

```
print(len_files * 2)
train = []
for i in range(0, len_files, self.__max_files):
    start = i
    end = i + self.__max_files if len_files > i + self.__max_files else len_files
    size = end - start
    print(start, end, size)
    train_x = np.zeros((size, ImageNet.size_img, ImageNet.size_img, 3), dtype='float32')
    train_y = np.zeros((size, ImageNet.size_img, ImageNet.size_img, 1), dtype='float32')
    j = 0
    for file in list_files[start:end]:
        orig = cv2.imread(os.path.join(orig_ds_dir, file))
        res = cv2.imread(os.path.join(res_ds_dir, file), 0)
        train_x[j] = np.array(img_as_float(orig))
        train_y[j] = np.asarray(img_as_float(res))[..., None].astype(dtype='float32', copy=False)
        j += 1
    train.append((train_x, train_y))
    self.signals.updateProgress.emit(end)

# net
image_net = ImageNet()
len_train = len(train)
for i in range(self.__epochs):
    for j in range(0, len_train):
        train_x, train_y = train[j]
        image_net.train(train_x, train_y, 1)
    self.signals.updateProgress.emit(len_files + (i + 1) * len_files)

# save weights
weights_dir = os.path.join(os.getcwd(), 'weights')
weights_file_name = os.path.join(weights_dir, 'image_net_{}.weights'.format(self.__id))

image_net.save_weights(weights_file_name)
print('learn ready')

# check net
image_net.load_weights(weights_file_name)
train_x, train_y = train[0]
predictions = image_net.predict(train_x)

self.signals.finished.emit()
```

ЛістингА.9. – МодульLine.py

```
import numpy as np
import cv2


class Line:
    def __init__(self, coords):
        self.__pts = np.array(coords, np.int32).reshape((- 1, 1, 2))
        self.__rect = cv2.boundingRect(self.__pts)
        self.__mask = None
        self.__count_mask_pixels = 0
        self.__color = (255, 255, 255)
        self.__final_size = (256, 256)
        self.__final_orig_size = (256, 256, 3)

    def __get_mask(self, shape):
        if self.__mask is not None:
            return self.__mask
        self.__mask = np.full((shape[0], shape[1]), 0, dtype=np.uint8)
        cv2.fillPoly(self.__mask, [self.__pts], self.__color)
        x, y, w, h = self.__rect
        self.__mask = self.__mask[y:y + h, x:x + w]
        self.__count_mask_pixels = cv2.countNonZero(self.__mask)
        return self.__mask

    def get_dataset_images(self, orig_image, res_image):
        return self.get_image(orig_image), self.get_image(res_image, False)
        # mask = self.__get_mask(orig_image.shape)
        # x, y, w, h = self.__rect
        #
        # orig_crop = orig_image[y:y + h, x:x + w]
        # final_orig_image = np.full(self.__final_orig_size, 0, dtype=np.uint8)
        # final_orig_image[0:h, 0:w] = cv2.bitwise_or(orig_crop, orig_crop, mask=mask)
        #
        # crop = res_image[y:y + h, x:x + w]
        # final_image = np.full(self.__final_size, 0, dtype=np.uint8)
        # final_image[0:h, 0:w] = cv2.bitwise_or(crop, crop, mask=mask)
        # return final_orig_image, final_image

    def get_image(self, image, orig=True):
        mask = self.__get_mask(image.shape)
```

```
        x, y, w, h = self.__rect
        orig_crop = image[y:y + h, x:x + w]
        final_image = np.full(self.__final_orig_size if orig is True else self.__final_size, 0, dtype=np.uint8)
        final_image[0:h, 0:w] = cv2.bitwise_or(orig_crop, orig_crop, mask=mask)
        return final_image

    def get_tlcr(self, image):
        return cv2.sumElems(image)[0] / self.__count_mask_pixels
```

ЛістингА.10. – МодульLoadVideoThread.py

```
from threading import Thread
import time
import os
import subprocess
import re
import urllib.parse
from classes.Video import Video
from PySide2 import QtCore


class LoadVideoThreadSignal(QtCore.QObject):
    updateProgress = QtCore.Signal(int)
    updateStatus = QtCore.Signal(str)
    updateProgressMax = QtCore.Signal(int)
    finished = QtCore.Signal()


class LoadVideoThread(Thread):
    __max_files = 200

    def __init__(self, camera_id, gplmod, server):
        super().__init__()
        self.__camera_id = camera_id
        self.__gplmod = gplmod
        self.__server = server
        self.signals = LoadVideoThreadSignal()

    def run(self):
        camera_dir = os.path.join(Video.video_folder, str(self.__camera_id))
        if not os.path.exists(camera_dir):
            os.makedirs(camera_dir)
        urls = Video.get_list_urls(self.__camera_id, self.__gplmod)
```

```
        size = len(urls)
        self.signals.updateProgressMax.emit(size)
        for i in range(0, size):
            self.signals.updateStatus.emit('Завантаження {}'.format(urls[i]))
            Video.download_video(urllib.parse.urljoin(self.__server, urls[i]), os.path.join(camera_dir, urls[i]))
            self.signals.updateProgress.emit(i + 1)

        self.signals.finished.emit()
```

ЛістингА.11. – МодульVideo.py

```
import requests
import os
import cv2
from PIL import Image

os.environ['OPENCV_FFMPEG_DEBUG'] = '0'
os.environ['OPENCV_VIDEOIO_DEBUG'] = '0'
# FFREPORT=level=quiet



class Video:
    video_folder = os.path.join(os.getcwd(), 'video')
    res_folder = 'res'
    orig_folder = 'orig'
    @staticmethod
    def get_list_urls(id, gplmod):
        url_gplmod = "http://videoprobki.ua/{}?p1=cam{}&p2={}&p3=1&p4=1".format(gplmod, id, 1)
        list_urls = requests.get(url_gplmod).text.split(", ")
        print('eba', url_gplmod, list_urls)
        list_urls.pop(0)
        # list_urls.pop()
        return list_urls

    @staticmethod
    def get_image(id, gplmod, server):
        list_urls = Video.get_list_urls(id, gplmod)
        print(list_urls)
        cam = cv2.VideoCapture(server + list_urls[0])
        # cam = cv2.VideoCapture('D:/projects/python/network/src/video/22/cam22stream_1594272662.mp4')
        ret, frame = cam.read()
        print(id, gplmod, server + list_urls[-1])
        if not ret:
```

```
            return
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        return frame
        # cv2.imshow('trololo', frame)
        # return Image.fromarray(frame)
        # Video.download_video(server + list_urls[-1], os.path.join(Video.video_folder, list_urls[-1]))

    @staticmethod
    def download_video(url, save_url):
        if os.path.isfile(save_url):
            return
        r = requests.get(url)
        with open(save_url, 'wb') as f:
            for chunk in r.iter_content(chunk_size=1024 * 1024):
                if chunk:
                    f.write(chunk)

    @staticmethod
    def fill_images(video_file, frames):
        cam = cv2.VideoCapture(video_file)
        frame_count = cam.get(cv2.CAP_PROP_FRAME_COUNT)
        num_frame = 0
        for i in range(int(frame_count)):
            ret, frame = cam.read()
            if not ret:
                break
            elif num_frame % 5 == 0:
                frames.append(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
num_frame += 1
```

ДОДАТОК Б

Список публікацій здобувача

1. Stetsenko, I.V., Stelmakh, O. (2020). Traffic Lane Congestion Ratio Evaluation by Video Data. Advances in Intelligent Systems and Computing, 1019, 172-181. Springer, Cham. ISSN 2194-5357. https://doi.org/10.1007/978-3-030-25741-5_18 (Scopus)

2. Stelmakh O.P., Stetsenko I.V., Velyhotskyi D.V. (2020). Information technology of video data processing for traffic intensity monitoring. *Control systems & computers*, 3 (287), 49-59.

3. Стеценко І.В., Стельмах О.П. (2020). Технологія визначення інтенсивності дорожнього руху за даними відеоряду. *Технічні науки та технології: науковий журнал*, 2, 116-125.

4. Стеценко І.В., Стельмах О.П. (2017). Програмний компонент визначення інтенсивності транспортних потоків. *Вісник Національного технічного університету України «КПІ імені І. Сікорського»*, 66, 94-100.

ДОДАТОК В

Фактичні та прогнозовані показники завантаженості

Таблиця 1. Отримані фактичні та прогнозовані показники завантаженості TLCR з 5-ти хвилинним усередненням

1.02		2.02		3.02		4.02		5.02		
Фактичне TLCR	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Час***
0.027252	0.031916	0.012863	0.018254	0.176178	0.181806	0.136821	0.137198	0.121419	0.088291	8:00
0.024866	0.02995	0.010089	0.017637	0.201745	0.172796	0.141675	0.138705	0.132551	0.12931	8:05
0.023983	0.027889	0.012213	0.015392	0.211747	0.207589	0.135909	0.147191	0.124291	0.135967	8:10
0.027836	0.026992	0.017326	0.016978	0.199023	0.205648	0.154348	0.137309	0.108752	0.122729	8:15
0.032133	0.030215	0.018542	0.021251	0.199555	0.186987	0.174289	0.164159	0.10681	0.106314	8:20
0.033844	0.034045	0.019653	0.022433	0.201938	0.19385	0.158227	0.185834	0.13703	0.107129	8:25
0.043635	0.035688	0.020432	0.023391	0.206276	0.196042	0.141948	0.155876	0.173801	0.145804	8:30
0.042115	0.044921	0.02059	0.024077	0.196637	0.199624	0.136913	0.1395	0.190268	0.187192	8:35
0.037282	0.043332	0.017847	0.024233	0.18431	0.187318	0.139529	0.137706	0.207213	0.195263	8:40
0.045658	0.038808	0.02372	0.021929	0.190065	0.178029	0.136184	0.143127	0.205019	0.205184	8:45
0.04651	0.046906	0.02615	0.026835	0.179243	0.190186	0.121652	0.137246	0.192704	0.193602	8:50
0.043001	0.047529	0.027089	0.029062	0.176239	0.174698	0.134687	0.119242	0.189571	0.181801	8:55
0.042528	0.044094	0.029757	0.029915	0.183041	0.174516	0.143683	0.140704	0.194545	0.184714	9:00
0.038826	0.043684	0.019906	0.032287	0.199156	0.185177	0.133042	0.149925	0.198002	0.191524	9:05
0.034564	0.040137	0.021298	0.023825	0.19979	0.200857	0.131535	0.132896	0.19282	0.192708	9:10
0.040724	0.036254	0.02516	0.024819	0.176404	0.193556	0.137351	0.132967	0.211199	0.185336	9:15
0.049165	0.042095	0.020393	0.028152	0.164863	0.167023	0.140378	0.140544	0.212409	0.20847	9:20
0.050831	0.050248	0.033679	0.024137	0.172499	0.160232	0.113806	0.143152	0.156299	0.199638	9:25
0.055579	0.051747	0.038674	0.035723	0.187854	0.174616	0.078006	0.108568	0.117723	0.133555	9:30
0.065151	0.056514	0.031159	0.040351	0.190537	0.192828	0.069624	0.073265	0.127219	0.105959	9:35
0.063157	0.066401	0.038506	0.033436	0.185409	0.188353	0.112769	0.069506	0.111098	0.12938	9:40
0.052903	0.063746	0.041718	0.040241	0.193	0.18146	0.160561	0.121614	0.10788	0.107851	9:45
0.057062	0.053138	0.036711	0.043157	0.198933	0.192236	0.172322	0.178676	0.116621	0.108384	9:50
0.061798	0.058052	0.032335	0.038405	0.19866	0.195062	0.171073	0.180363	0.097855	0.120073	9:55

0.058534	0.062811	0.028812	0.034527	0.165969	0.192206	0.182045	0.171721	0.09053	0.09535	10:00
0.058105	0.058828	0.026878	0.031423	0.097673	0.153256	0.199692	0.18499	0.081503	0.091049	10:05
0.05797	0.058415	0.026721	0.02972	0.094901	0.081199	0.201524	0.200314	0.083568	0.081782	10:10
0.052431	0.058257	0.034788	0.029582	0.113785	0.093971	0.198894	0.193426	0.094721	0.0853	10:15
0.056083	0.052674	0.042397	0.036796	0.134086	0.117186	0.194825	0.190218	0.093506	0.098092	10:20
0.073334	0.056769	0.043582	0.043832	0.169392	0.138928	0.20243	0.189992	0.082733	0.095168	10:25
0.071646	0.074901	0.04955	0.044798	0.150774	0.180585	0.243465	0.198392	0.079103	0.083578	10:30
0.060915	0.071312	0.049234	0.050577	0.114549	0.143478	0.24225	0.236297	0.086896	0.080593	10:35
0.060413	0.060196	0.046503	0.050029	0.15269	0.104187	0.209574	0.221045	0.086515	0.089368	10:40
0.059762	0.060656	0.053622	0.04739	0.157001	0.163411	0.199507	0.186999	0.088362	0.087696	10:45
0.064244	0.059959	0.05815	0.054511	0.103565	0.157737	0.178121	0.185956	0.086884	0.089407	10:50
0.071097	0.064627	0.054463	0.05872	0.083241	0.089454	0.185208	0.167498	0.078455	0.087587	10:55
0.073649	0.071555	0.047665	0.054687	0.075967	0.079517	0.209928	0.186318	0.082089	0.078597	11:00
0.068004	0.073625	0.038247	0.048235	0.073621	0.074882	0.193764	0.209846	0.095544	0.083407	11:05
0.064632	0.067269	0.034064	0.039649	0.068659	0.073279	0.171351	0.179952	0.100321	0.098431	11:10
0.060925	0.064323	0.042981	0.036071	0.06952	0.068204	0.143502	0.161203	0.094685	0.101423	11:15
0.063001	0.060762	0.05336	0.044438	0.067597	0.069519	0.107328	0.133081	0.095137	0.093181	11:20
0.075077	0.06323	0.047775	0.054394	0.064643	0.067373	0.093639	0.097523	0.088223	0.094647	11:25
0.076323	0.07602	0.04254	0.04841	0.066887	0.064487	0.085746	0.090446	0.096534	0.086512	11:30
0.074109	0.076112	0.04489	0.043649	0.070147	0.067127	0.084028	0.083977	0.106314	0.097352	11:35
0.073534	0.073524	0.04964	0.04608	0.068437	0.070363	0.082866	0.083277	0.095295	0.107599	11:40
0.066355	0.073162	0.054703	0.050627	0.062567	0.068195	0.088616	0.082245	0.096172	0.092512	11:45
0.060156	0.065552	0.061186	0.055469	0.069076	0.062205	0.094789	0.088937	0.117027	0.095732	11:50
0.061921	0.059856	0.060296	0.061809	0.08119	0.069575	0.093235	0.095242	0.119874	0.121045	11:55
0.061884	0.062206	0.052593	0.06043	0.076267	0.082257	0.093607	0.092372	0.102035	0.11989	12:00
0.062893	0.061997	0.05454	0.052724	0.072206	0.075357	0.098701	0.093079	0.126531	0.097413	12:05
0.059487	0.063039	0.054454	0.055154	0.073549	0.071559	0.101283	0.098987	0.258726	0.132091	12:10
0.062479	0.059455	0.053873	0.054943	0.071454	0.073462	0.111133	0.10111	0.285994	0.284299	12:15
0.062059	0.062819	0.058585	0.054364	0.071313	0.07106	0.109979	0.112565	0.180354	0.255655	12:20
0.062967	0.062119	0.053592	0.059174	0.088633	0.071136	0.090236	0.109048	0.138213	0.137756	12:25
0.071004	0.063101	0.048837	0.053803	0.094158	0.090487	0.07076	0.085986	0.113152	0.123011	12:30
0.068114	0.071583	0.054551	0.049423	0.076846	0.0945	0.070745	0.068091	0.121202	0.10585	12:35

0.067614	0.067695	0.055969	0.055343	0.076842	0.074105	0.083259	0.070543	0.119056	0.122475	12:40
0.065093	0.067454	0.055528	0.056469	0.079015	0.076482	0.07636	0.084434	0.08391	0.117713	12:45
0.058586	0.064836	0.060635	0.055951	0.081118	0.078878	0.063036	0.075213	0.084385	0.077069	12:50
0.067582	0.058328	0.062114	0.061173	0.094552	0.080958	0.059437	0.061924	0.096583	0.083941	12:55
0.079855	0.068253	0.05265	0.062323	0.096537	0.096105	0.068322	0.05948	0.086626	0.097973	13:00
0.071052	0.080931	0.053744	0.052622	0.089474	0.096269	0.071966	0.069073	0.087525	0.08442	13:05
0.062808	0.069819	0.056723	0.054343	0.088475	0.087726	0.066227	0.072189	0.086106	0.087089	13:10
0.069441	0.062099	0.053494	0.057266	0.096817	0.08774	0.062672	0.065695	0.08239	0.085331	13:15
0.061911	0.069878	0.061247	0.053851	0.099504	0.097596	0.060834	0.06253	0.093967	0.081381	13:20
0.061904	0.061339	0.06597	0.061952	0.104669	0.09931	0.067667	0.060932	0.098238	0.095243	13:25
0.061679	0.061998	0.05563	0.066348	0.109034	0.104987	0.06758	0.06829	0.092933	0.098369	13:30
0.049735	0.061765	0.052797	0.055369	0.110128	0.10928	0.062417	0.067559	0.084718	0.091369	13:35
0.058728	0.049771	0.057723	0.053232	0.112343	0.109668	0.062873	0.062135	0.079245	0.082927	13:40
0.066665	0.059544	0.049873	0.05834	0.104659	0.112132	0.066303	0.063048	0.078915	0.078078	13:45
0.059001	0.067295	0.052686	0.050226	0.083088	0.10233	0.075243	0.066559	0.079439	0.078453	13:50
0.055469	0.058631	0.064866	0.053444	0.067405	0.079069	0.072619	0.075883	0.08655	0.079093	13:55
0.051123	0.055663	0.061975	0.065807	0.073429	0.065565	0.06869	0.072039	0.090475	0.087044	14:00
0.046063	0.051559	0.06207	0.061921	0.071165	0.073783	0.064938	0.068126	0.099392	0.090505	14:05
0.051286	0.046851	0.061355	0.062229	0.064922	0.070712	0.062448	0.064589	0.097145	0.100336	14:10
0.05025	0.052183	0.050557	0.061489	0.069649	0.064364	0.066827	0.062326	0.084273	0.096047	14:15
0.048278	0.050911	0.053143	0.050708	0.061764	0.069963	0.073019	0.067148	0.09735	0.081683	14:20
0.052236	0.049041	0.056984	0.053874	0.060332	0.061251	0.076317	0.07342	0.109211	0.098994	14:25
0.054604	0.053045	0.064467	0.057596	0.065152	0.060414	0.08456	0.076359	0.103993	0.111065	14:30
0.052906	0.055211	0.071275	0.065095	0.05607	0.065531	0.079237	0.085178	0.090594	0.102159	14:35
0.056842	0.053376	0.071502	0.071736	0.066083	0.055798	0.066604	0.078071	0.10168	0.087555	14:40
0.062736	0.057441	0.070733	0.071286	0.07212	0.066833	0.075405	0.065166	0.110941	0.103041	14:45
0.062055	0.063259	0.068418	0.070436	0.062961	0.072495	0.076892	0.076028	0.095939	0.112222	14:50
0.062688	0.062084	0.070625	0.06804	0.060662	0.062215	0.072268	0.076701	0.089233	0.092291	14:55
0.067601	0.062811	0.069888	0.070621	0.066666	0.060647	0.072888	0.071479	0.081864	0.087532	15:00
0.062704	0.067949	0.066682	0.069621	0.066274	0.06711	0.079359	0.072682	0.075661	0.080333	15:05
0.061096	0.062345	0.073929	0.066305	0.060794	0.06617	0.101191	0.079738	0.077303	0.074597	15:10
0.064418	0.061128	0.067801	0.074398	0.072735	0.060509	0.111733	0.104267	0.069822	0.077275	15:15

0.05701	0.064804	0.058821	0.067081	0.081008	0.073646	0.095158	0.113311	0.062069	0.068843	15:20
0.056862	0.056878	0.058225	0.058387	0.068202	0.081607	0.083504	0.091207	0.06338	0.06151	15:25
0.063097	0.057216	0.05507	0.058472	0.064923	0.066711	0.087442	0.081163	0.064101	0.063567	15:30
0.062866	0.063654	0.063162	0.05532	0.063287	0.064615	0.092702	0.08746	0.065057	0.064199	15:35
0.062869	0.062925	0.065254	0.063828	0.062371	0.063209	0.086205	0.092931	0.076974	0.065128	15:40
0.064961	0.062967	0.061683	0.065409	0.068969	0.062411	0.088773	0.084601	0.076554	0.077948	15:45
0.061819	0.065131	0.060119	0.061515	0.068728	0.069435	0.098317	0.088577	0.069162	0.076186	15:50
0.066014	0.061657	0.058136	0.060186	0.070481	0.068581	0.090951	0.099318	0.068978	0.068196	15:55
0.061408	0.066318	0.058275	0.058291	0.065471	0.070475	0.079277	0.089031	0.059358	0.068794	16:00
0.054917	0.061125	0.061976	0.058594	0.066017	0.06498	0.080443	0.077162	0.05852	0.058777	16:05
0.059468	0.054912	0.057022	0.062378	0.078849	0.06605	0.084429	0.080127	0.064637	0.058732	16:10
0.062585	0.060071	0.053555	0.05702	0.074026	0.079921	0.084038	0.084451	0.068046	0.065147	16:15
0.057672	0.062923	0.057818	0.053902	0.065761	0.073173	0.07899	0.083449	0.066905	0.068255	16:20
0.054179	0.057626	0.057273	0.058423	0.069026	0.06493	0.077686	0.077863	0.060558	0.066744	16:25
0.062401	0.054442	0.057466	0.057574	0.068141	0.0692	0.077839	0.077111	0.057124	0.060254	16:30
0.066478	0.063089	0.058247	0.057798	0.06917	0.067949	0.06997	0.077461	0.059992	0.057256	16:35
0.060933	0.066762	0.057501	0.058598	0.069298	0.069133	0.076875	0.068922	0.064644	0.060396	16:40
0.049673	0.060614	0.052499	0.057824	0.061763	0.069181	0.077896	0.07732	0.062144	0.065024	16:45
0.055048	0.049788	0.057639	0.052849	0.058995	0.06128	0.069444	0.077643	0.063874	0.062031	16:50
0.072162	0.055817	0.060315	0.058284	0.066875	0.059083	0.072826	0.068366	0.071286	0.064048	16:55
0.110686	0.073476	0.06609	0.060716	0.076493	0.067486	0.06385	0.072931	0.072258	0.07179	17:00
0.100304	0.116962	0.068934	0.066543	0.083534	0.07719	0.060976	0.062991	0.08961	0.072101	17:05
0.057677	0.09749	0.053092	0.069076	0.07269	0.084023	0.061933	0.06088	0.094341	0.091452	17:10
0.062652	0.05325	0.052512	0.052542	0.071982	0.071138	0.055233	0.062107	0.07021	0.0945	17:15
0.060378	0.063186	0.058948	0.05309	0.072312	0.071671	0.047873	0.0552	0.070976	0.067036	17:20
0.053508	0.060644	0.060028	0.059645	0.062613	0.072098	0.042656	0.04842	0.077185	0.07092	17:25
0.052309	0.05402	0.05654	0.060345	0.062109	0.061825	0.044614	0.04377	0.067088	0.077587	17:30
0.053331	0.053146	0.058979	0.056713	0.064305	0.062204	0.070768	0.0458	0.067732	0.065899	17:35
0.056766	0.054209	0.063707	0.059416	0.070006	0.06452	0.132075	0.07256	0.070183	0.067846	17:40
0.053419	0.057605	0.062764	0.064152	0.071302	0.070367	0.157	0.144494	0.062172	0.070661	17:45
0.048174	0.053954	0.05772	0.062824	0.064651	0.071218	0.161892	0.164215	0.056634	0.061905	17:50
0.055563	0.048924	0.058344	0.057698	0.05938	0.064062	0.163772	0.162631	0.060246	0.05682	17:55

0.059978	0.056598	0.05933	0.05868	0.06588	0.059211	0.185905	0.16335	0.06404	0.061011	18:00
0.051884	0.060957	0.056845	0.059672	0.06565	0.06639	0.194964	0.193043	0.063107	0.06486	18:05
0.054672	0.052421	0.055746	0.057064	0.066671	0.065623	0.191796	0.193339	0.06873	0.06362	18:10
0.060684	0.05556	0.04968	0.056101	0.076026	0.06673	0.187312	0.18513	0.069999	0.069683	18:15
0.05532	0.061721	0.054516	0.050172	0.073657	0.076726	0.179121	0.181795	0.066791	0.070535	18:20
0.052367	0.055858	0.059763	0.055299	0.069	0.07311	0.189795	0.174365	0.066409	0.067071	18:25
0.057332	0.053073	0.051592	0.060358	0.114349	0.068385	0.178002	0.191177	0.05858	0.066815	18:30
0.054516	0.058247	0.056338	0.0518	0.166632	0.121882	0.198206	0.173983	0.056579	0.058746	18:35
0.052405	0.055214	0.067393	0.057039	0.177622	0.182352	0.188385	0.20426	0.060183	0.057115	18:40
0.050737	0.053102	0.062091	0.068247	0.173875	0.180522	0.157043	0.183138	0.058213	0.060805	18:45
0.044243	0.051409	0.061251	0.06183	0.17027	0.171138	0.170308	0.145688	0.05922	0.058606	18:50
0.045936	0.045137	0.060645	0.061395	0.194908	0.167741	0.172803	0.178175	0.062037	0.059791	18:55
0.048412	0.04698	0.052453	0.060826	0.219979	0.201158	0.187044	0.178259	0.056709	0.06276	19:00
0.049631	0.0494	0.055456	0.052607	0.207132	0.216035	0.196085	0.196402	0.053291	0.057098	19:05
0.051657	0.050502	0.058267	0.056095	0.182042	0.190002	0.167857	0.197319	0.058381	0.053785	19:10
0.050577	0.052421	0.053526	0.058775	0.179421	0.168782	0.111443	0.158339	0.058472	0.058963	19:15
0.047249	0.051225	0.047549	0.053835	0.202963	0.177218	0.051339	0.096838	0.058585	0.058786	19:20
0.053824	0.048041	0.051809	0.048219	0.208151	0.205718	0.04071	0.046259	0.063875	0.058948	19:25
0.050317	0.054702	0.048182	0.052688	0.186259	0.199599	0.050719	0.041741	0.060347	0.06436	19:30
0.03169	0.050919	0.040828	0.049019	0.113503	0.173423	0.050656	0.051765	0.057761	0.060296	19:35
0.029295	0.03375	0.043928	0.04217	0.045587	0.092434	0.04876	0.051419	0.062791	0.057938	19:40
0.037515	0.031898	0.046293	0.045272	0.041698	0.040404	0.051241	0.049583	0.05647	0.063286	19:45
0.039261	0.039341	0.045573	0.047456	0.041804	0.042985	0.053526	0.052102	0.049054	0.05643	19:50
0.04154	0.040908	0.04619	0.046675	0.04354	0.043215	0.049253	0.054249	0.051732	0.049536	19:55
0.044989	0.043012	0.043953	0.047284	0.046204	0.044865	0.045219	0.049904	0.056001	0.05256	20:00
0.039539	0.046244	0.045383	0.045138	0.041699	0.04735	0.048045	0.046223	0.05773	0.056693	20:05
0.043574	0.041006	0.051774	0.046561	0.038565	0.042956	0.051092	0.049115	0.054413	0.058189	20:10
0.050422	0.044922	0.048513	0.052739	0.039453	0.040167	0.053609	0.051996	0.05052	0.054723	20:15
0.051897	0.051443	0.043748	0.049246	0.039174	0.041063	0.049309	0.05436	0.049596	0.051084	20:20
0.047914	0.052637	0.043925	0.044822	0.038916	0.040802	0.04078	0.04999	0.051417	0.050412	20:25
0.047317	0.048666	0.043412	0.045156	0.042124	0.040554	0.047363	0.042023	0.047141	0.052218	20:30
0.053601	0.048265	0.041	0.044639	0.04089	0.043578	0.052456	0.048573	0.04328	0.047933	20:35

0.058782	0.054485	0.034226	0.042354	0.038395	0.042374	0.041539	0.05338	0.040637	0.044456	20:40

Таблиця 2. Отримані фактичні та прогнозовані показники завантаженості TLCR з 10-ти хвилинним усередненням

1.02		**2.02**		**3.02**		**4.02**		**5.02**		
Фактичне TLCR	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Час***
0.027252	0.031916	0.012863	0.018254	0.176178	0.181806	0.136821	0.137198	0.121419	0.088291	8:00
0.023983	0.027889	0.012213	0.015392	0.211747	0.207589	0.135909	0.147191	0.124291	0.135967	8:10
0.032133	0.030215	0.018542	0.021251	0.199555	0.186987	0.174289	0.164159	0.10681	0.106314	8:20
0.043635	0.035688	0.020432	0.023391	0.206276	0.196042	0.141948	0.155876	0.173801	0.145804	8:30
0.037282	0.043332	0.017847	0.024233	0.18431	0.187318	0.139529	0.137706	0.207213	0.195263	8:40
0.04651	0.046906	0.02615	0.026835	0.179243	0.190186	0.121652	0.137246	0.192704	0.193602	8:50
0.042528	0.044094	0.029757	0.029915	0.183041	0.174516	0.143683	0.140704	0.194545	0.184714	9:00
0.034564	0.040137	0.021298	0.023825	0.19979	0.200857	0.131535	0.132896	0.19282	0.192708	9:10
0.049165	0.042095	0.020393	0.028152	0.164863	0.167023	0.140378	0.140544	0.212409	0.20847	9:20
0.055579	0.051747	0.038674	0.035723	0.187854	0.174616	0.078006	0.108568	0.117723	0.133555	9:30
0.063157	0.066401	0.038506	0.033436	0.185409	0.188353	0.112769	0.069506	0.111098	0.12938	9:40
0.057062	0.053138	0.036711	0.043157	0.198933	0.192236	0.172322	0.178676	0.116621	0.108384	9:50
0.058534	0.062811	0.028812	0.034527	0.165969	0.192206	0.182045	0.171721	0.09053	0.09535	10:00
0.05797	0.058415	0.026721	0.02972	0.094901	0.081199	0.201524	0.200314	0.083568	0.081782	10:10
0.056083	0.052674	0.042397	0.036796	0.134086	0.117186	0.194825	0.190218	0.093506	0.098092	10:20
0.071646	0.074901	0.04955	0.044798	0.150774	0.180585	0.243465	0.198392	0.079103	0.083578	10:30
0.060413	0.060196	0.046503	0.050029	0.15269	0.104187	0.209574	0.221045	0.086515	0.089368	10:40
0.064244	0.059959	0.05815	0.054511	0.103565	0.157737	0.178121	0.185956	0.086884	0.089407	10:50
0.073649	0.071555	0.047665	0.054687	0.075967	0.079517	0.209928	0.186318	0.082089	0.078597	11:00
0.064632	0.067269	0.034064	0.039649	0.068659	0.073279	0.171351	0.179952	0.100321	0.098431	11:10
0.063001	0.060762	0.05336	0.044438	0.067597	0.069519	0.107328	0.133081	0.095137	0.093181	11:20
0.076323	0.07602	0.04254	0.04841	0.066887	0.064487	0.085746	0.090446	0.096534	0.086512	11:30
0.073534	0.073524	0.04964	0.04608	0.068437	0.070363	0.082866	0.083277	0.095295	0.107599	11:40
0.060156	0.065552	0.061186	0.055469	0.069076	0.062205	0.094789	0.088937	0.117027	0.095732	11:50

0.061884	0.062206	0.052593	0.06043	0.076267	0.082257	0.093607	0.092372	0.102035	0.11989	12:00
0.059487	0.063039	0.054454	0.055154	0.073549	0.071559	0.101283	0.098987	0.258726	0.132091	12:10
0.062059	0.062819	0.058585	0.054364	0.071313	0.07106	0.109979	0.112565	0.180354	0.255655	12:20
0.071004	0.063101	0.048837	0.053803	0.094158	0.090487	0.07076	0.085986	0.113152	0.123011	12:30
0.067614	0.067695	0.055969	0.055343	0.076842	0.074105	0.083259	0.070543	0.119056	0.122475	12:40
0.058586	0.064836	0.060635	0.055951	0.081118	0.078878	0.063036	0.075213	0.084385	0.077069	12:50
0.079855	0.068253	0.05265	0.062323	0.096537	0.096105	0.068322	0.05948	0.086626	0.097973	13:00
0.062808	0.069819	0.056723	0.054343	0.088475	0.087726	0.066227	0.072189	0.086106	0.087089	13:10
0.061911	0.069878	0.061247	0.053851	0.099504	0.097596	0.060834	0.06253	0.093967	0.081381	13:20
0.061679	0.061998	0.05563	0.066348	0.109034	0.104987	0.06758	0.06829	0.092933	0.098369	13:30
0.058728	0.049771	0.057723	0.053232	0.112343	0.109668	0.062873	0.062135	0.079245	0.082927	13:40
0.059001	0.067295	0.052686	0.050226	0.083088	0.10233	0.075243	0.066559	0.079439	0.078453	13:50
0.051123	0.055663	0.061975	0.065807	0.073429	0.065565	0.06869	0.072039	0.090475	0.087044	14:00
0.051286	0.046851	0.061355	0.062229	0.064922	0.070712	0.062448	0.064589	0.097145	0.100336	14:10
0.048278	0.050911	0.053143	0.050708	0.061764	0.069963	0.073019	0.067148	0.09735	0.081683	14:20
0.054604	0.053045	0.064467	0.057596	0.065152	0.060414	0.08456	0.076359	0.103993	0.111065	14:30
0.056842	0.053376	0.071502	0.071736	0.066083	0.055798	0.066604	0.078071	0.10168	0.087555	14:40
0.062055	0.063259	0.068418	0.070436	0.062961	0.072495	0.076892	0.076028	0.095939	0.112222	14:50
0.067601	0.062811	0.069888	0.070621	0.066666	0.060647	0.072888	0.071479	0.081864	0.087532	15:00
0.061096	0.062345	0.073929	0.066305	0.060794	0.06617	0.101191	0.079738	0.077303	0.074597	15:10
0.05701	0.064804	0.058821	0.067081	0.081008	0.073646	0.095158	0.113311	0.062069	0.068843	15:20
0.063097	0.057216	0.05507	0.058472	0.064923	0.066711	0.087442	0.081163	0.064101	0.063567	15:30
0.062869	0.062925	0.065254	0.063828	0.062371	0.063209	0.086205	0.092931	0.076974	0.065128	15:40
0.061819	0.065131	0.060119	0.061515	0.068728	0.069435	0.098317	0.088577	0.069162	0.076186	15:50
0.061408	0.066318	0.058275	0.058291	0.065471	0.070475	0.079277	0.089031	0.059358	0.068794	16:00
0.059468	0.054912	0.057022	0.062378	0.078849	0.06605	0.084429	0.080127	0.064637	0.058732	16:10
0.057672	0.062923	0.057818	0.053902	0.065761	0.073173	0.07899	0.083449	0.066905	0.068255	16:20
0.062401	0.054442	0.057466	0.057574	0.068141	0.0692	0.077839	0.077111	0.057124	0.060254	16:30
0.060933	0.066762	0.057501	0.058598	0.069298	0.069133	0.076875	0.068922	0.064644	0.060396	16:40
0.055048	0.049788	0.057639	0.052849	0.058995	0.06128	0.069444	0.077643	0.063874	0.062031	16:50
0.110686	0.073476	0.06609	0.060716	0.076493	0.067486	0.06385	0.072931	0.072258	0.07179	17:00
0.057677	0.09749	0.053092	0.069076	0.07269	0.084023	0.061933	0.06088	0.094341	0.091452	17:10

0.060378	0.063186	0.058948	0.05309	0.072312	0.071671	0.047873	0.0552	0.070976	0.067036	17:20
0.052309	0.05402	0.05654	0.060345	0.062109	0.061825	0.044614	0.04377	0.067088	0.077587	17:30
0.056766	0.054209	0.063707	0.059416	0.070006	0.06452	0.132075	0.07256	0.070183	0.067846	17:40
0.048174	0.053954	0.05772	0.062824	0.064651	0.071218	0.161892	0.164215	0.056634	0.061905	17:50
0.059978	0.056598	0.05933	0.05868	0.06588	0.059211	0.185905	0.16335	0.06404	0.061011	18:00
0.054672	0.052421	0.055746	0.057064	0.066671	0.065623	0.191796	0.193339	0.06873	0.06362	18:10
0.05532	0.061721	0.054516	0.050172	0.073657	0.076726	0.179121	0.181795	0.066791	0.070535	18:20
0.057332	0.053073	0.051592	0.060358	0.114349	0.068385	0.178002	0.191177	0.05858	0.066815	18:30
0.052405	0.055214	0.067393	0.057039	0.177622	0.182352	0.188385	0.20426	0.060183	0.057115	18:40
0.044243	0.051409	0.061251	0.06183	0.17027	0.171138	0.170308	0.145688	0.05922	0.058606	18:50
0.048412	0.04698	0.052453	0.060826	0.219979	0.201158	0.187044	0.178259	0.056709	0.06276	19:00
0.051657	0.050502	0.058267	0.056095	0.182042	0.190002	0.167857	0.197319	0.058381	0.053785	19:10
0.047249	0.051225	0.047549	0.053835	0.202963	0.177218	0.051339	0.096838	0.058585	0.058786	19:20
0.050317	0.054702	0.048182	0.052688	0.186259	0.199599	0.050719	0.041741	0.060347	0.06436	19:30
0.029295	0.03375	0.043928	0.04217	0.045587	0.092434	0.04876	0.051419	0.062791	0.057938	19:40
0.039261	0.039341	0.045573	0.047456	0.041804	0.042985	0.053526	0.052102	0.049054	0.05643	19:50
0.044989	0.043012	0.043953	0.047284	0.046204	0.044865	0.045219	0.049904	0.056001	0.05256	20:00
0.043574	0.041006	0.051774	0.046561	0.038565	0.042956	0.051092	0.049115	0.054413	0.058189	20:10
0.051897	0.051443	0.043748	0.049246	0.039174	0.041063	0.049309	0.05436	0.049596	0.051084	20:20
0.047317	0.048666	0.043412	0.045156	0.042124	0.040554	0.047363	0.042023	0.047141	0.052218	20:30
0.058782	0.054485	0.034226	0.042354	0.038395	0.042374	0.041539	0.05338	0.040637	0.044456	20:40

Таблиця 3. Отримані фактичні та прогнозовані показники завантаженості TLCR з 15-ти хвилинним усередненням

1.02		**2.02**		**3.02**		**4.02**		**5.02**		
Фактичне TLCR	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Фактичне TLCR***	***Прогнозоване TLCR***	***Час***
0.027252	0.031916	0.012863	0.018254	0.176178	0.181806	0.136821	0.137198	0.121419	0.088291	8:00
0.027836	0.026992	0.017326	0.016978	0.199023	0.205648	0.154348	0.137309	0.108752	0.122729	8:15
0.043635	0.035688	0.020432	0.023391	0.206276	0.196042	0.141948	0.155876	0.173801	0.145804	8:30
0.045658	0.038808	0.02372	0.021929	0.190065	0.178029	0.136184	0.143127	0.205019	0.205184	8:45
0.042528	0.044094	0.029757	0.029915	0.183041	0.174516	0.143683	0.140704	0.194545	0.184714	9:00

0.040724	0.036254	0.02516	0.024819	0.176404	0.193556	0.137351	0.132967	0.211199	0.185336	9:15
0.055579	0.051747	0.038674	0.035723	0.187854	0.174616	0.078006	0.108568	0.117723	0.133555	9:30
0.052903	0.063746	0.041718	0.040241	0.193	0.18146	0.160561	0.121614	0.10788	0.107851	9:45
0.058534	0.062811	0.028812	0.034527	0.165969	0.192206	0.182045	0.171721	0.09053	0.09535	10:00
0.052431	0.058257	0.034788	0.029582	0.113785	0.093971	0.198894	0.193426	0.094721	0.0853	10:15
0.071646	0.074901	0.04955	0.044798	0.150774	0.180585	0.243465	0.198392	0.079103	0.083578	10:30
0.059762	0.060656	0.053622	0.04739	0.157001	0.163411	0.199507	0.186999	0.088362	0.087696	10:45
0.073649	0.071555	0.047665	0.054687	0.075967	0.079517	0.209928	0.186318	0.082089	0.078597	11:00
0.060925	0.064323	0.042981	0.036071	0.06952	0.068204	0.143502	0.161203	0.094685	0.101423	11:15
0.076323	0.07602	0.04254	0.04841	0.066887	0.064487	0.085746	0.090446	0.096534	0.086512	11:30
0.066355	0.073162	0.054703	0.050627	0.062567	0.068195	0.088616	0.082245	0.096172	0.092512	11:45
0.061884	0.062206	0.052593	0.06043	0.076267	0.082257	0.093607	0.092372	0.102035	0.11989	12:00
0.062479	0.059455	0.053873	0.054943	0.071454	0.073462	0.111133	0.10111	0.285994	0.284299	12:15
0.071004	0.063101	0.048837	0.053803	0.094158	0.090487	0.07076	0.085986	0.113152	0.123011	12:30
0.065093	0.067454	0.055528	0.056469	0.079015	0.076482	0.07636	0.084434	0.08391	0.117713	12:45
0.079855	0.068253	0.05265	0.062323	0.096537	0.096105	0.068322	0.05948	0.086626	0.097973	13:00
0.069441	0.062099	0.053494	0.057266	0.096817	0.08774	0.062672	0.065695	0.08239	0.085331	13:15
0.061679	0.061998	0.05563	0.066348	0.109034	0.104987	0.06758	0.06829	0.092933	0.098369	13:30
0.066665	0.059544	0.049873	0.05834	0.104659	0.112132	0.066303	0.063048	0.078915	0.078078	13:45
0.051123	0.055663	0.061975	0.065807	0.073429	0.065565	0.06869	0.072039	0.090475	0.087044	14:00
0.05025	0.052183	0.050557	0.061489	0.069649	0.064364	0.066827	0.062326	0.084273	0.096047	14:15
0.054604	0.053045	0.064467	0.057596	0.065152	0.060414	0.08456	0.076359	0.103993	0.111065	14:30
0.062736	0.057441	0.070733	0.071286	0.07212	0.066833	0.075405	0.065166	0.110941	0.103041	14:45
0.067601	0.062811	0.069888	0.070621	0.066666	0.060647	0.072888	0.071479	0.081864	0.087532	15:00
0.064418	0.061128	0.067801	0.074398	0.072735	0.060509	0.111733	0.104267	0.069822	0.077275	15:15
0.063097	0.057216	0.05507	0.058472	0.064923	0.066711	0.087442	0.081163	0.064101	0.063567	15:30
0.064961	0.062967	0.061683	0.065409	0.068969	0.062411	0.088773	0.084601	0.076554	0.077948	15:45
0.061408	0.066318	0.058275	0.058291	0.065471	0.070475	0.079277	0.089031	0.059358	0.068794	16:00
0.062585	0.060071	0.053555	0.05702	0.074026	0.079921	0.084038	0.084451	0.068046	0.065147	16:15
0.062401	0.054442	0.057466	0.057574	0.068141	0.0692	0.077839	0.077111	0.057124	0.060254	16:30
0.049673	0.060614	0.052499	0.057824	0.061763	0.069181	0.077896	0.07732	0.062144	0.065024	16:45
0.110686	0.073476	0.06609	0.060716	0.076493	0.067486	0.06385	0.072931	0.072258	0.07179	17:00

0.062652	0.05325	0.052512	0.052542	0.071982	0.071138	0.055233	0.062107	0.07021	0.0945	17:15
0.052309	0.05402	0.05654	0.060345	0.062109	0.061825	0.044614	0.04377	0.067088	0.077587	17:30
0.053419	0.057605	0.062764	0.064152	0.071302	0.070367	0.157	0.144494	0.062172	0.070661	17:45
0.059978	0.056598	0.05933	0.05868	0.06588	0.059211	0.185905	0.16335	0.06404	0.061011	18:00
0.060684	0.05556	0.04968	0.056101	0.076026	0.06673	0.187312	0.18513	0.069999	0.069683	18:15
0.057332	0.053073	0.051592	0.060358	0.114349	0.068385	0.178002	0.191177	0.05858	0.066815	18:30
0.050737	0.053102	0.062091	0.068247	0.173875	0.180522	0.157043	0.183138	0.058213	0.060805	18:45
0.048412	0.04698	0.052453	0.060826	0.219979	0.201158	0.187044	0.178259	0.056709	0.06276	19:00
0.050577	0.052421	0.053526	0.058775	0.179421	0.168782	0.111443	0.158339	0.058472	0.058963	19:15
0.050317	0.054702	0.048182	0.052688	0.186259	0.199599	0.050719	0.041741	0.060347	0.06436	19:30
0.037515	0.031898	0.046293	0.045272	0.041698	0.040404	0.051241	0.049583	0.05647	0.063286	19:45
0.044989	0.043012	0.043953	0.047284	0.046204	0.044865	0.045219	0.049904	0.056001	0.05256	20:00
0.050422	0.044922	0.048513	0.052739	0.039453	0.040167	0.053609	0.051996	0.05052	0.054723	20:15
0.047317	0.048666	0.043412	0.045156	0.042124	0.040554	0.047363	0.042023	0.047141	0.052218	20:30

ДОДАТОК Г

Графічне відображення фактичних та прогнозованих показників завантаженості TLCR для різних днів тижня з 5-ти, 10-ти та 15-ти хвилинним усередненням

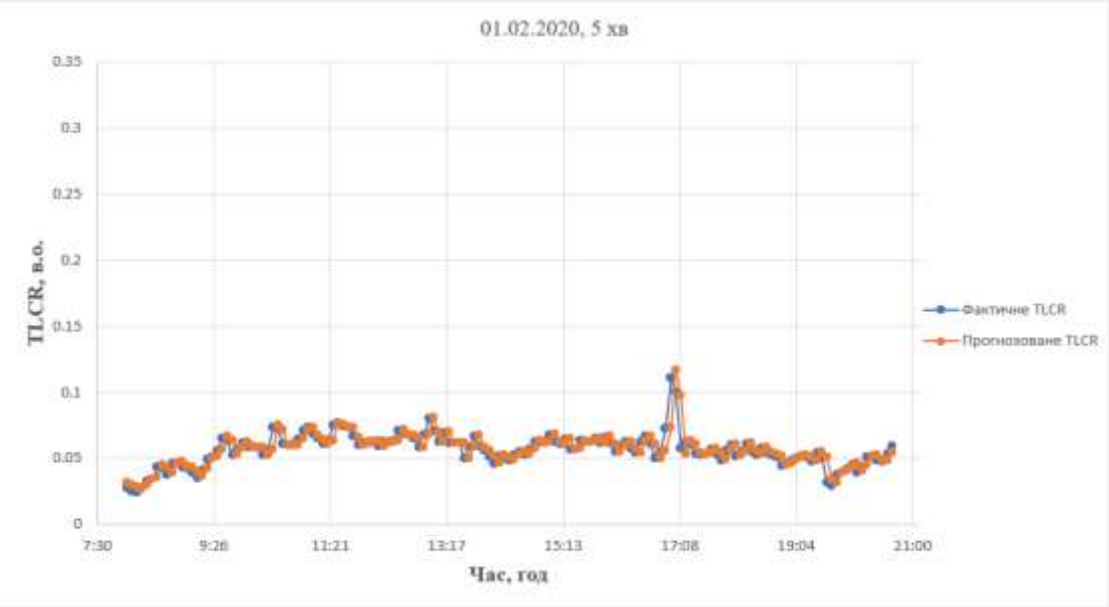


Рис. 1. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у суботу з 5-ти хвилинним усередненням

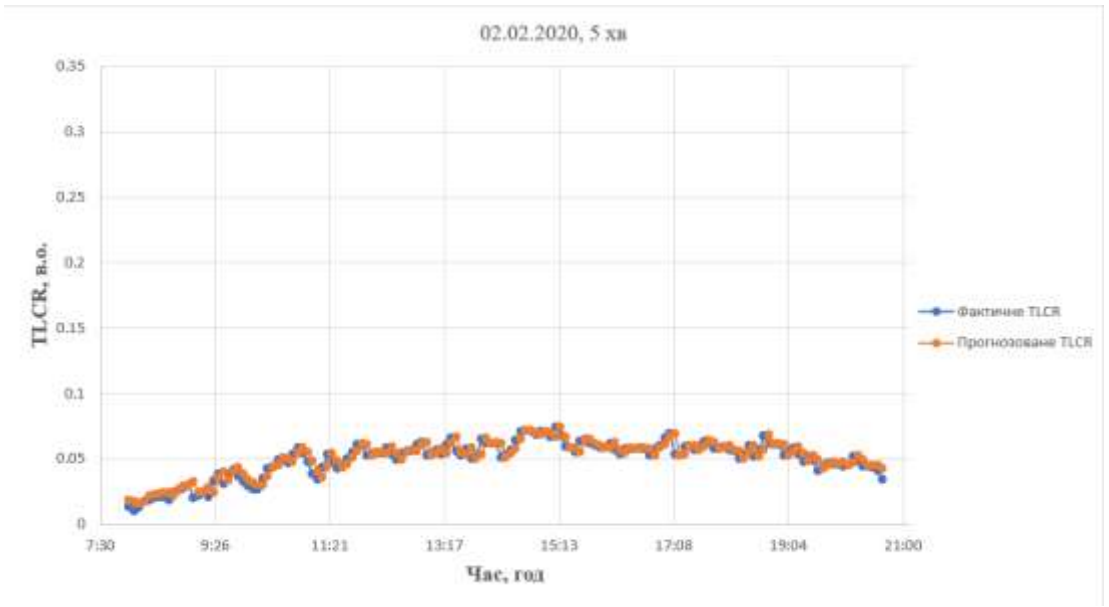


Рис. 2. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у неділю з 5-ти хвилинним усередненням

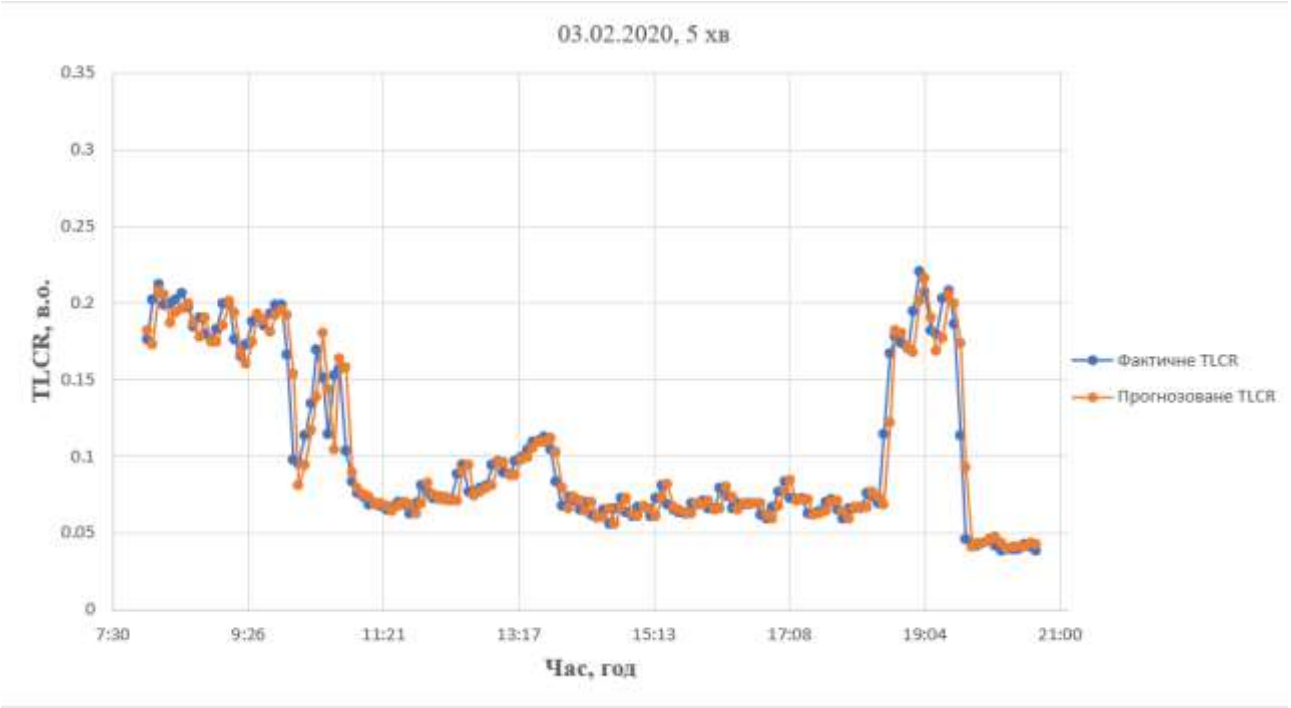


Рис. 3. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) в понеділок з 5-ти хвилинним усередненням

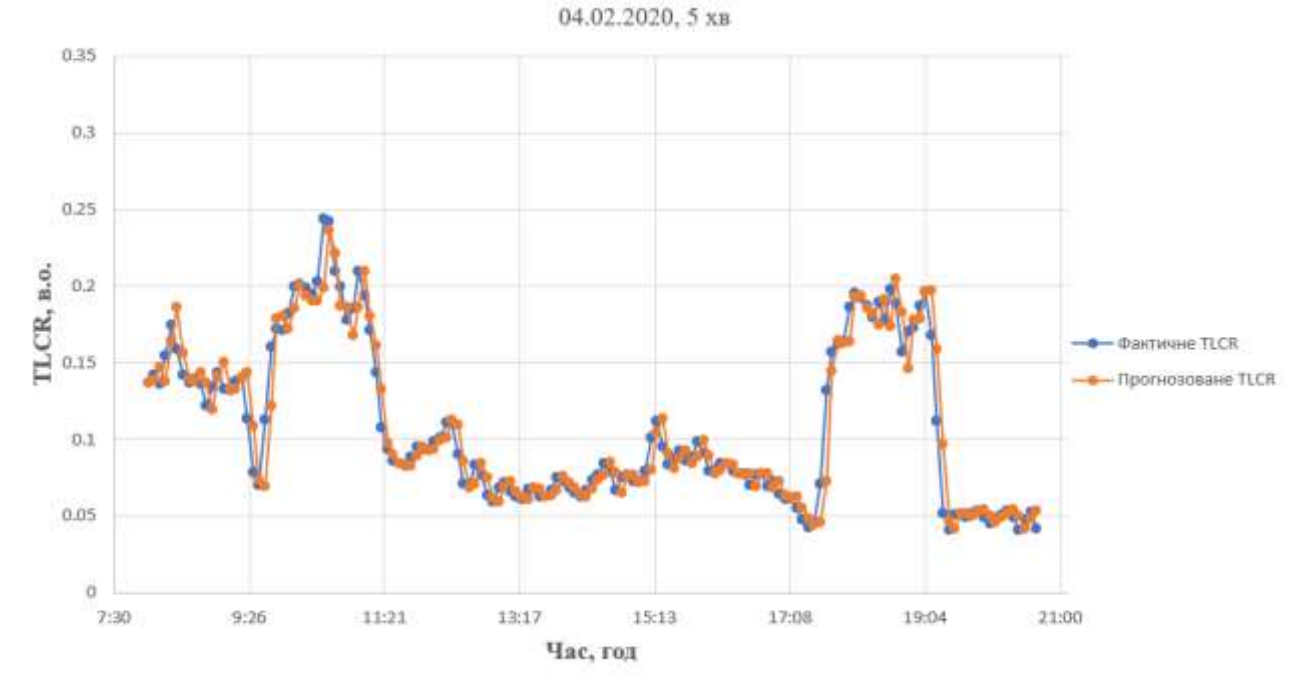


Рис. 4. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у вівторок з 5-ти хвилинним усередненням

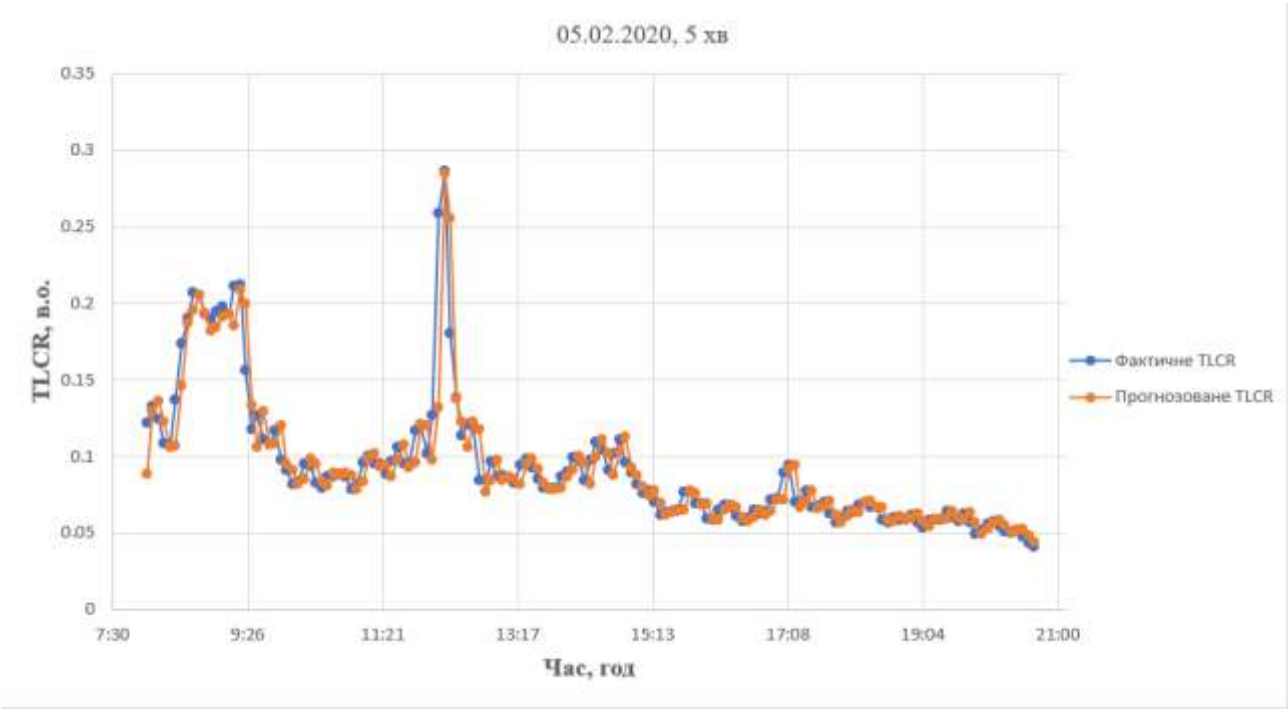


Рис. 5. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у середу з 5-ти хвилинним усередненням

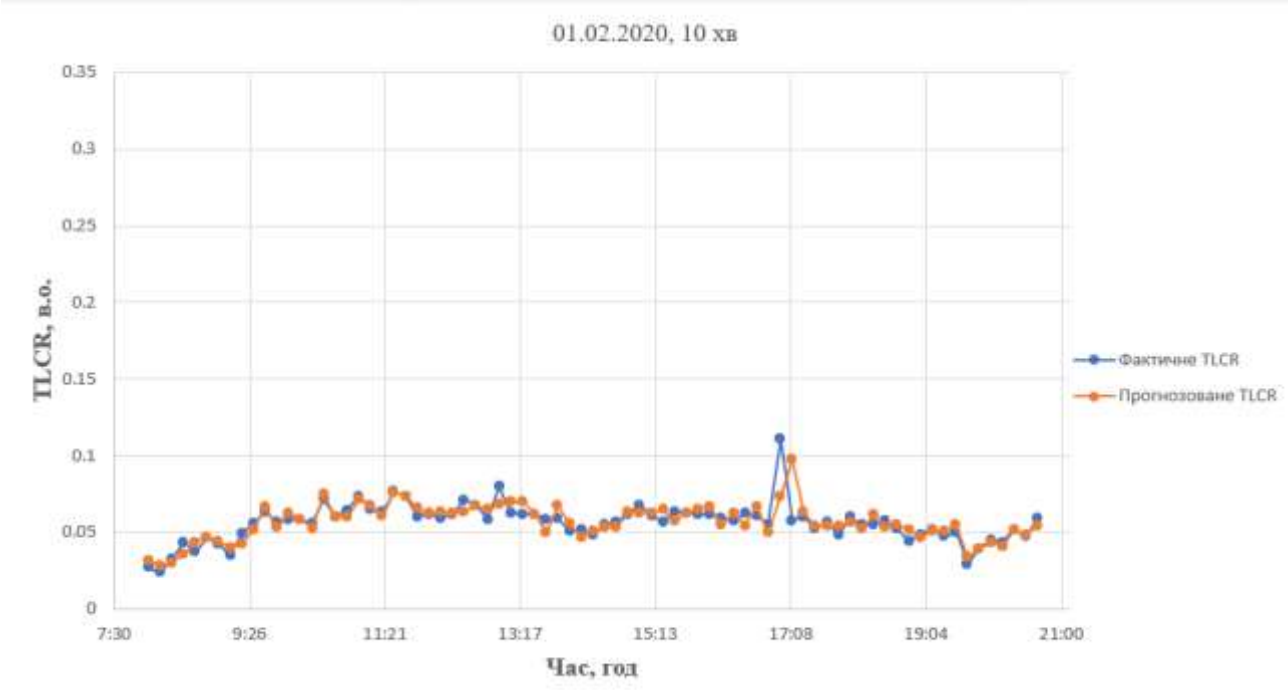


Рис. 6. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у суботу з 10-ти хвилинним усередненням

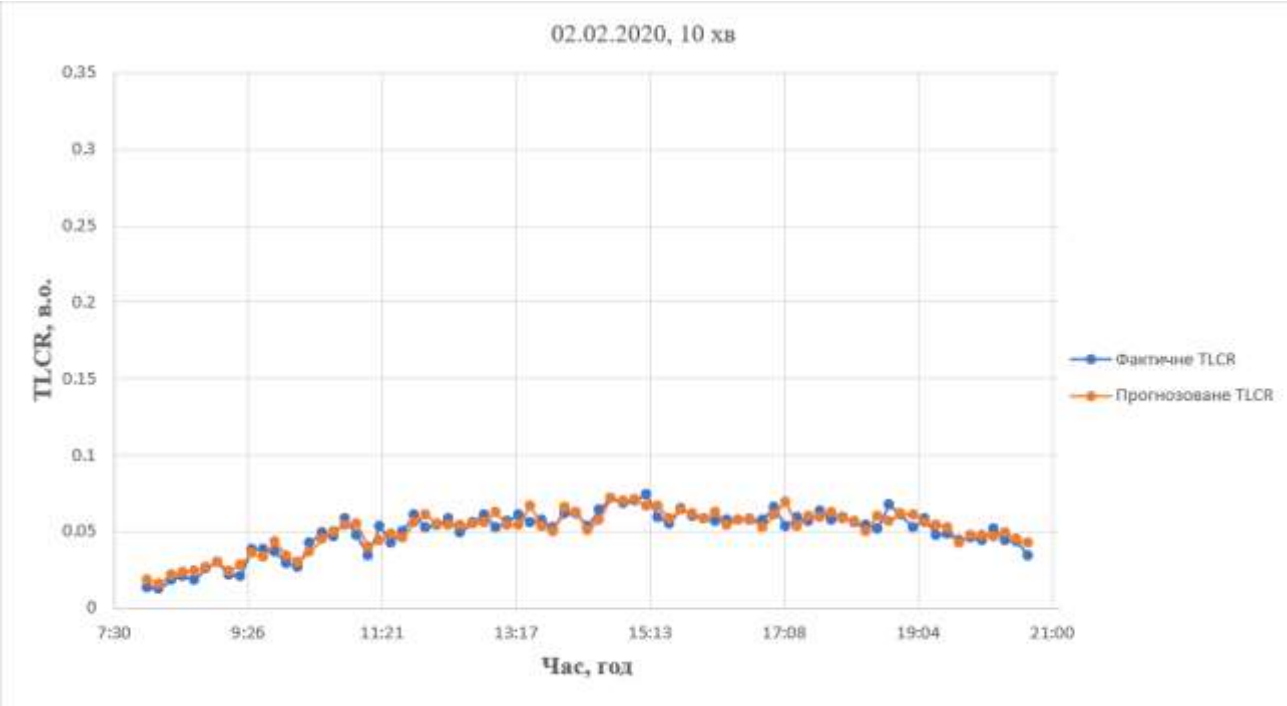


Рис. 7. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у неділю з 10-ти хвилинним усередненням

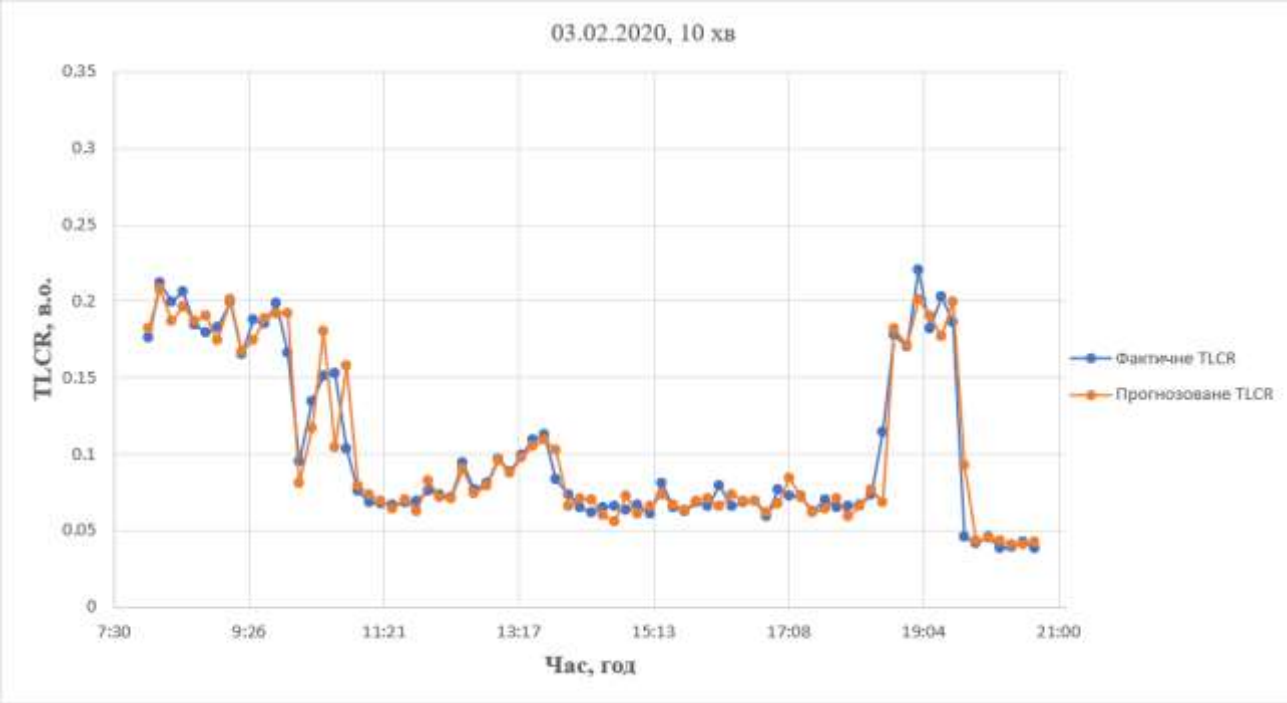


Рис. 8. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) в понеділок з 10-ти хвилинним усередненням

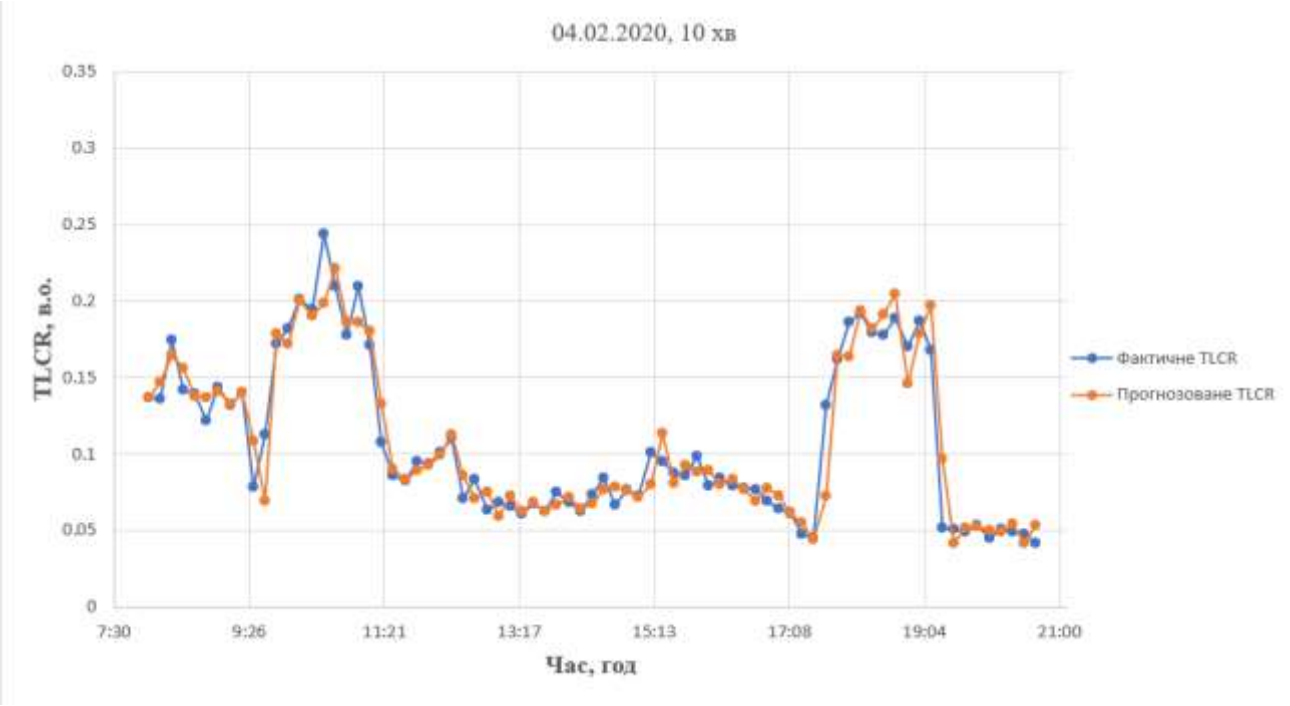


Рис. 9. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у вівторок з 10-ти хвилинним усередненням

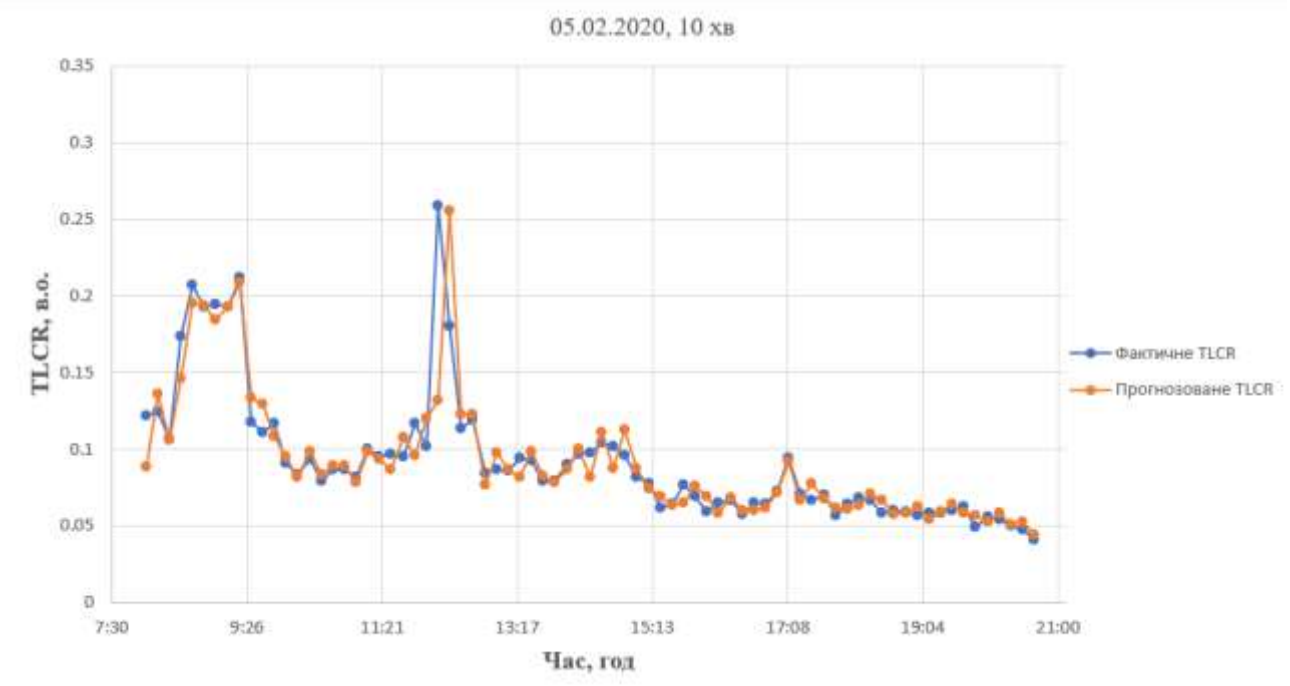


Рис. 10. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у середу з 10-ти хвилинним усередненням

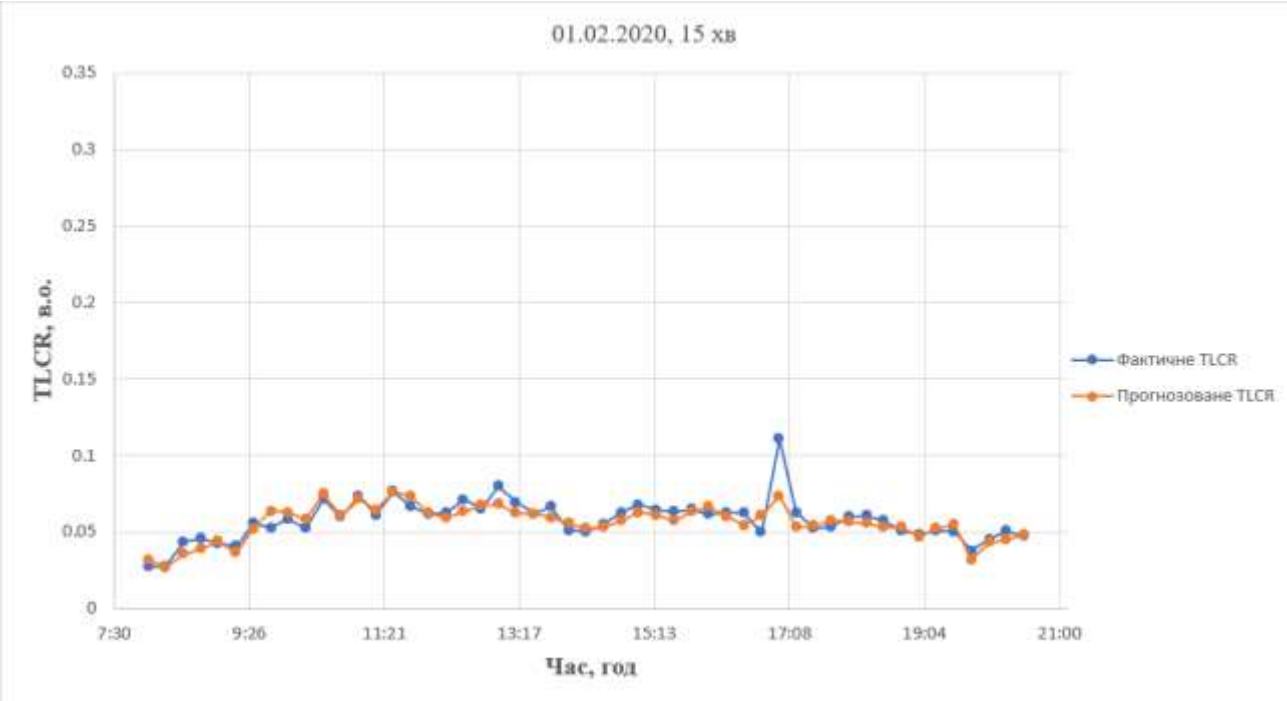


Рис. 1. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у суботу з 15-ти хвилинним усередненням

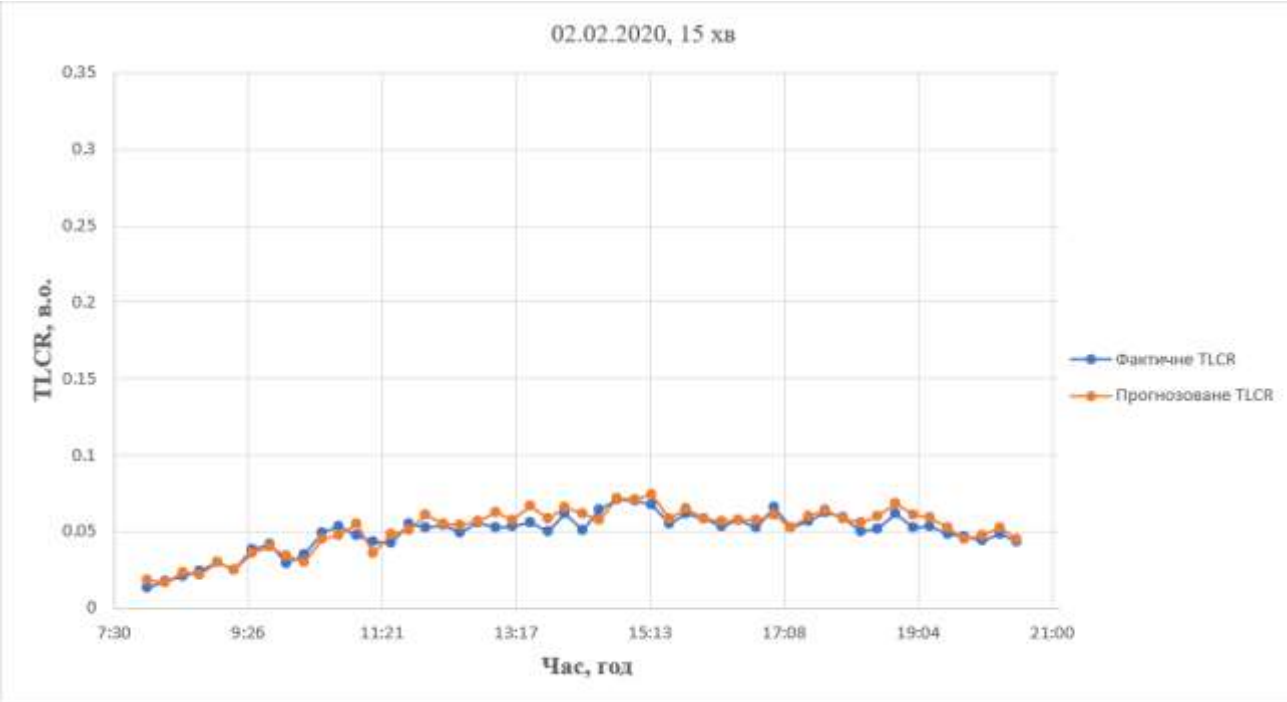


Рис. 12. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у неділю з 15-ти хвилинним усередненням

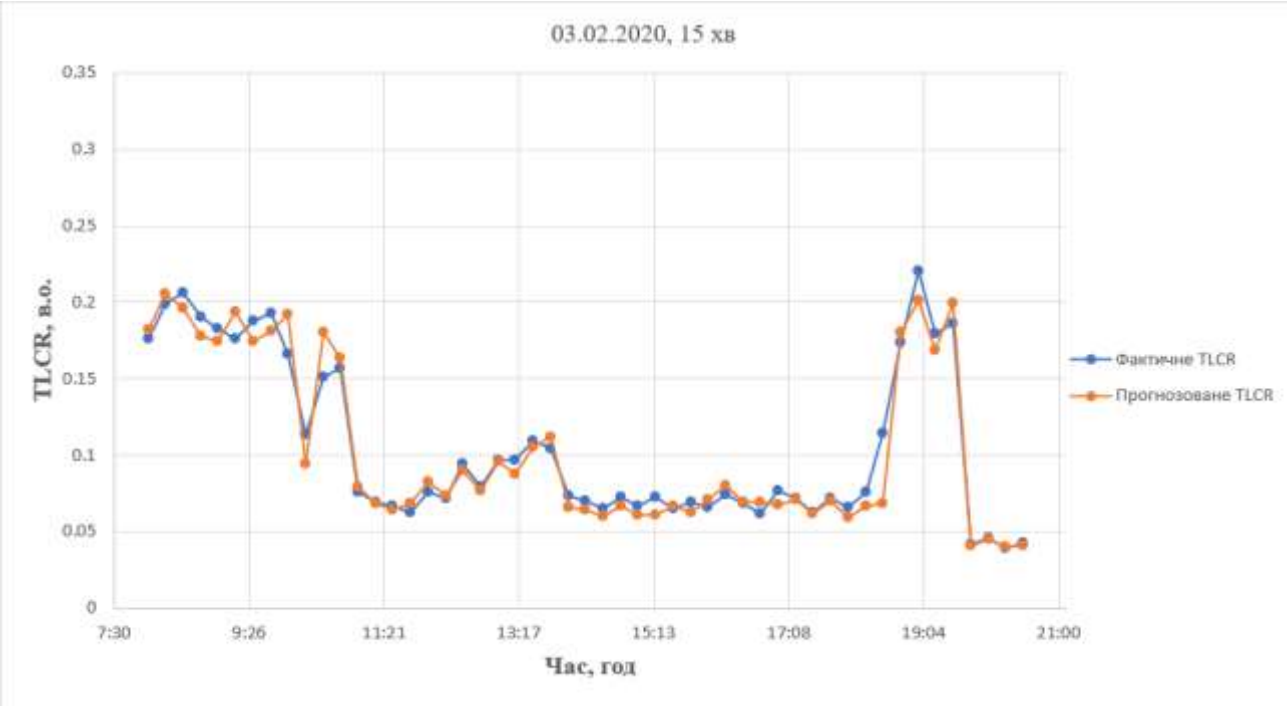


Рис. 13. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) в понеділок з 15-ти хвилинним усередненням

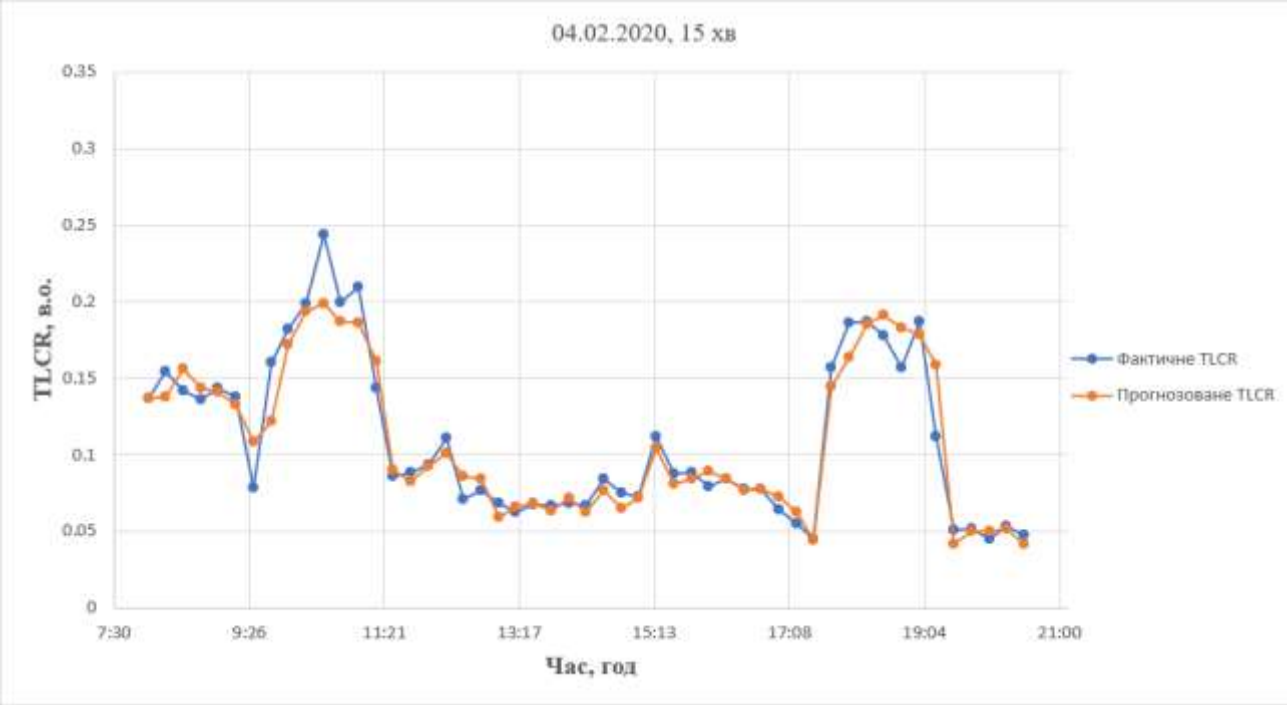


Рис. 14. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) у вівторок з 15-ти хвилинним усередненням

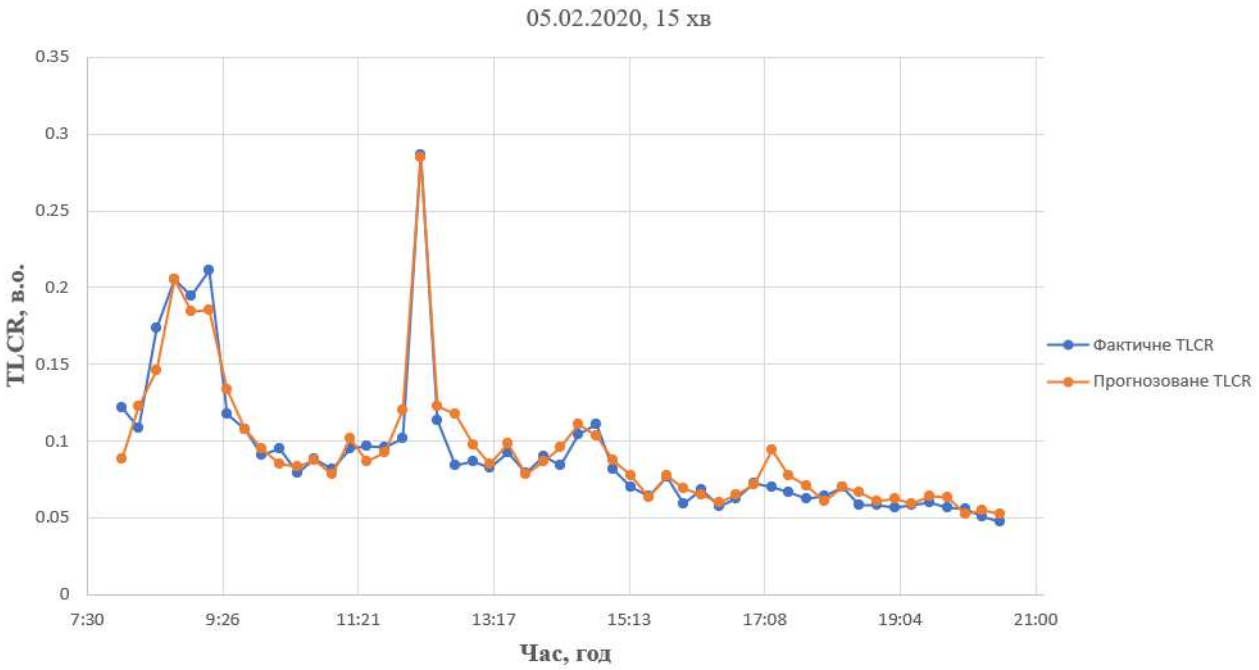


Рис. 15. Отримане прогнозування за показником завантаженості TLCR (помаранчева лінія) та реальне TLCR (синя лінія) в середу з 15-ти хвилинним усередненням

ДОДАТОК Д

Графіки абсолютної похибки прогнозування показника завантаженості TLCR для різних днів тижня

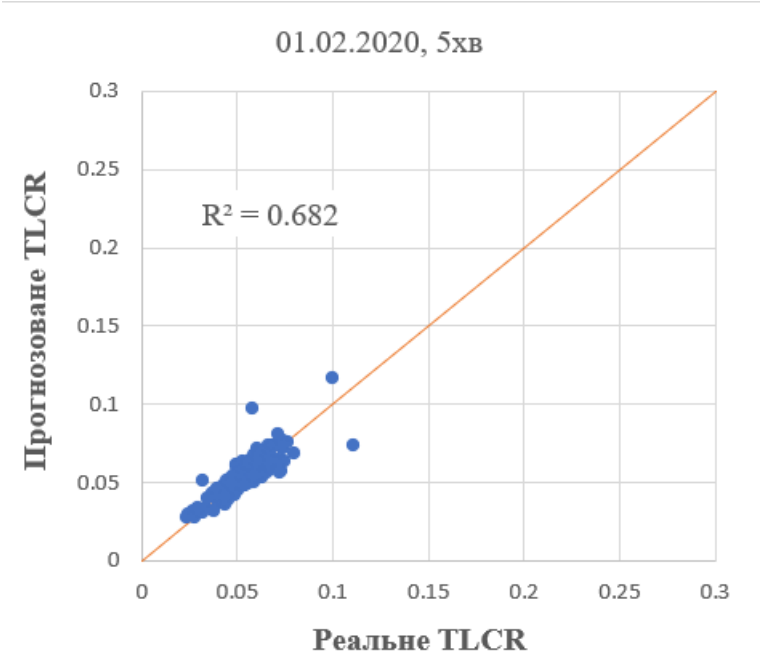


Рис. 1. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для суботи з 5-ти хвилинним усередненням

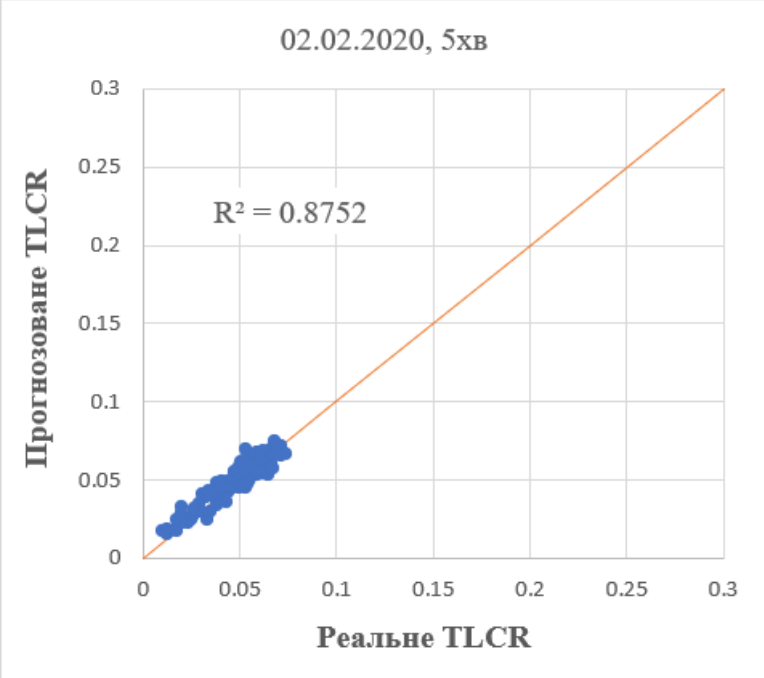


Рис. 2. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для неділі з 5-ти хвилинним усередненням

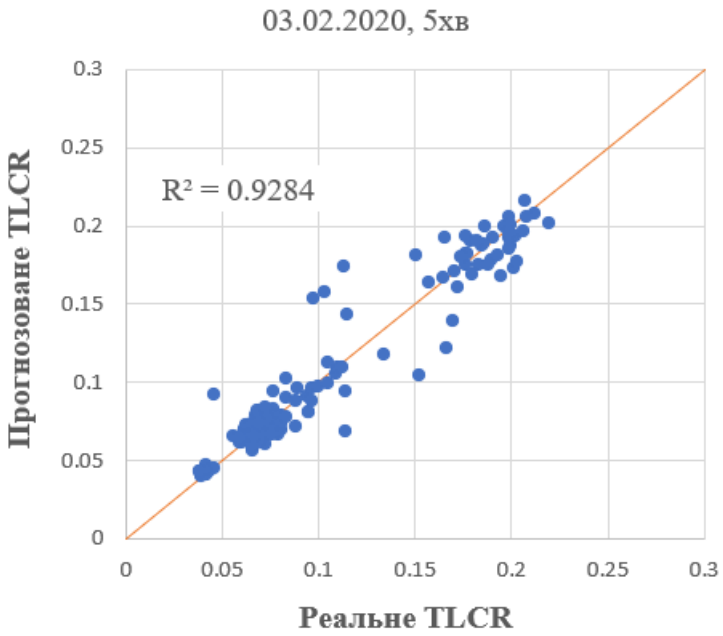


Рис. 3. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для понеділка з 5-ти хвилинним усередненням

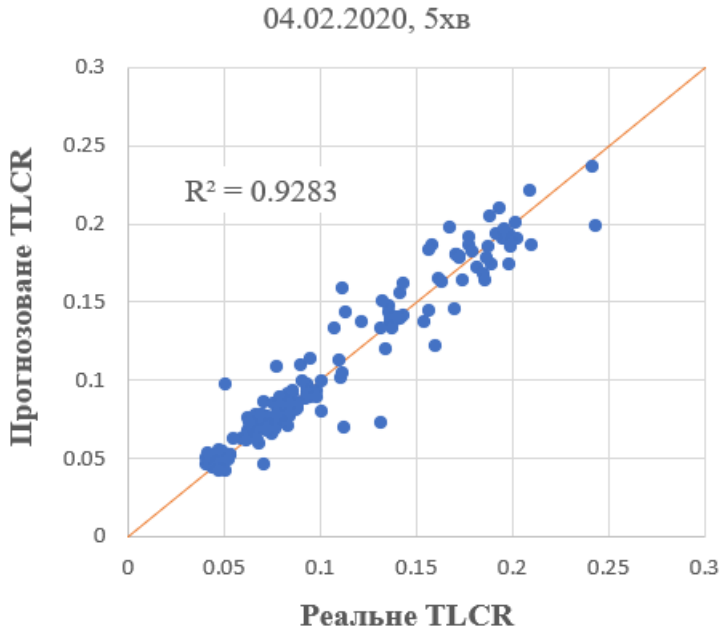


Рис. 4. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для вівторка з 5-ти хвилинним усередненням

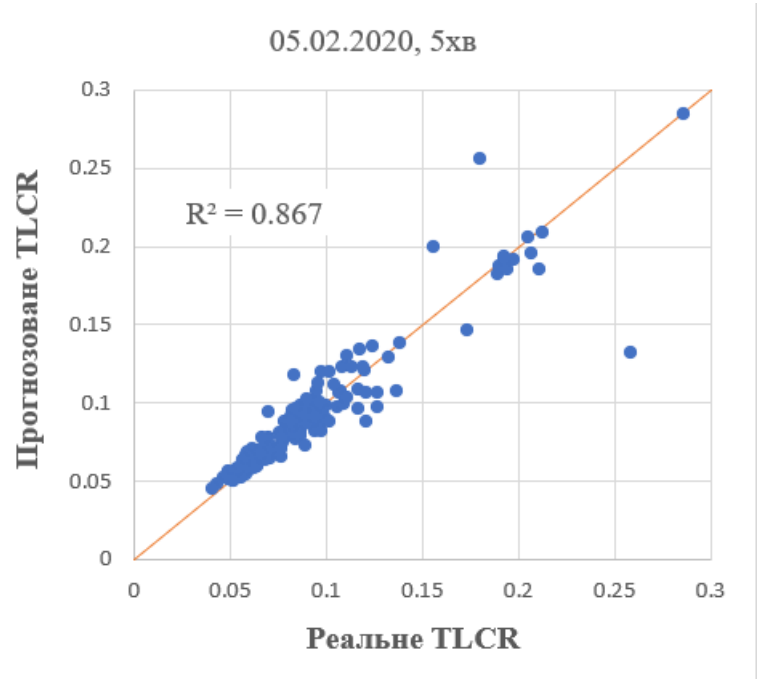


Рис. 5. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для середи з 5-ти хвилинним усередненням

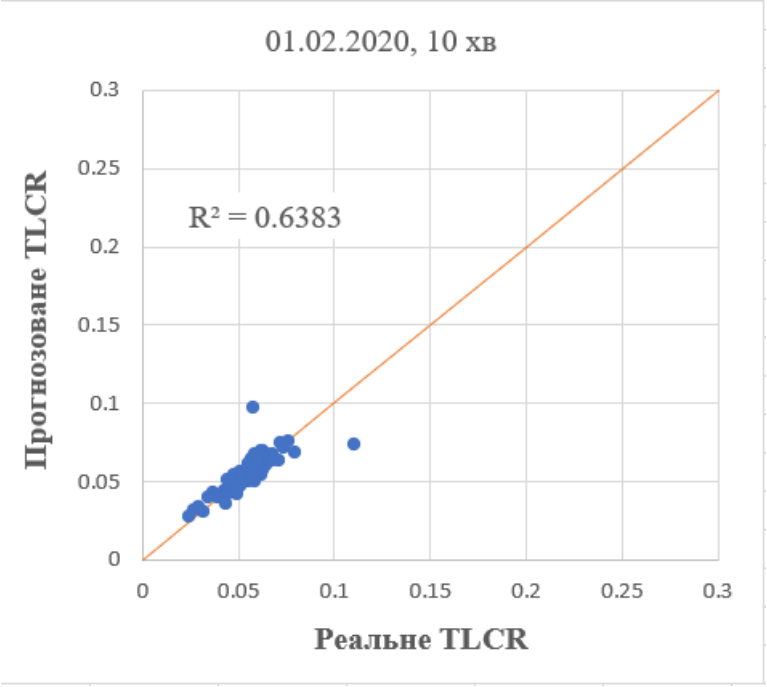


Рис. 6. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для суботи з 10-ти хвилинним усередненням

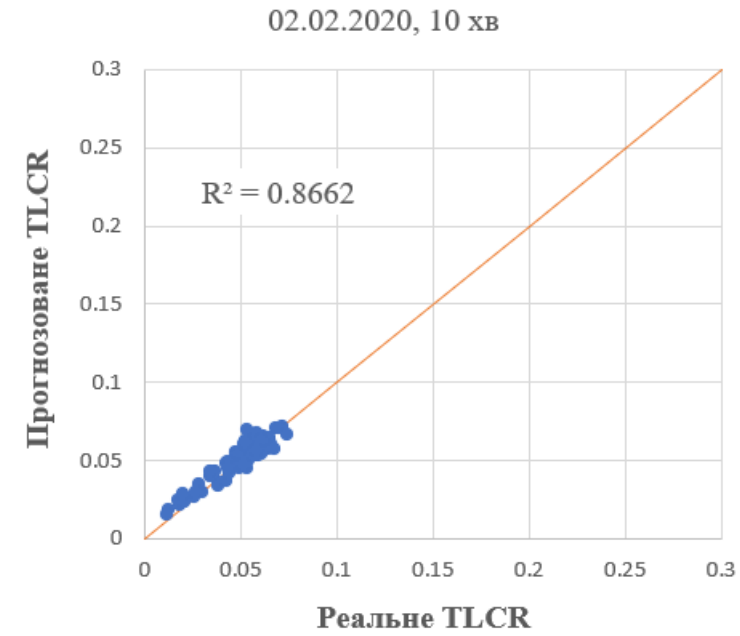


Рис. 7. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для неділі з 10-ти хвилинним усередненням

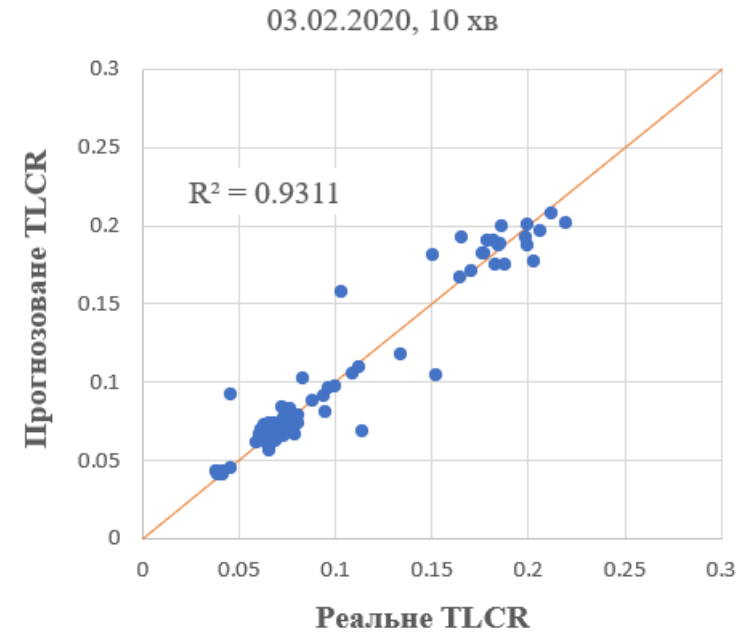


Рис. 8. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для понеділка з 10-ти хвилинним усередненням

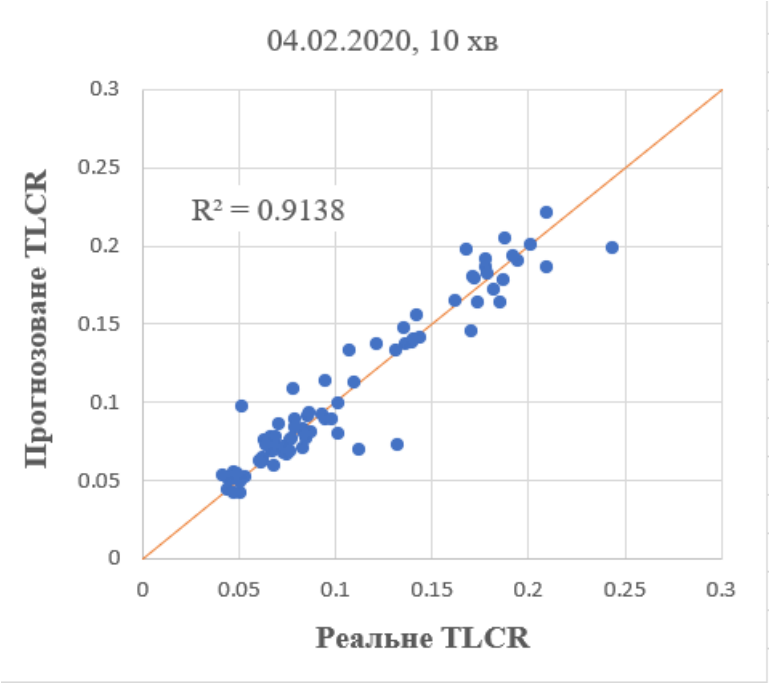


Рис. 9. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для вівторка з 10-ти хвилинним усередненням

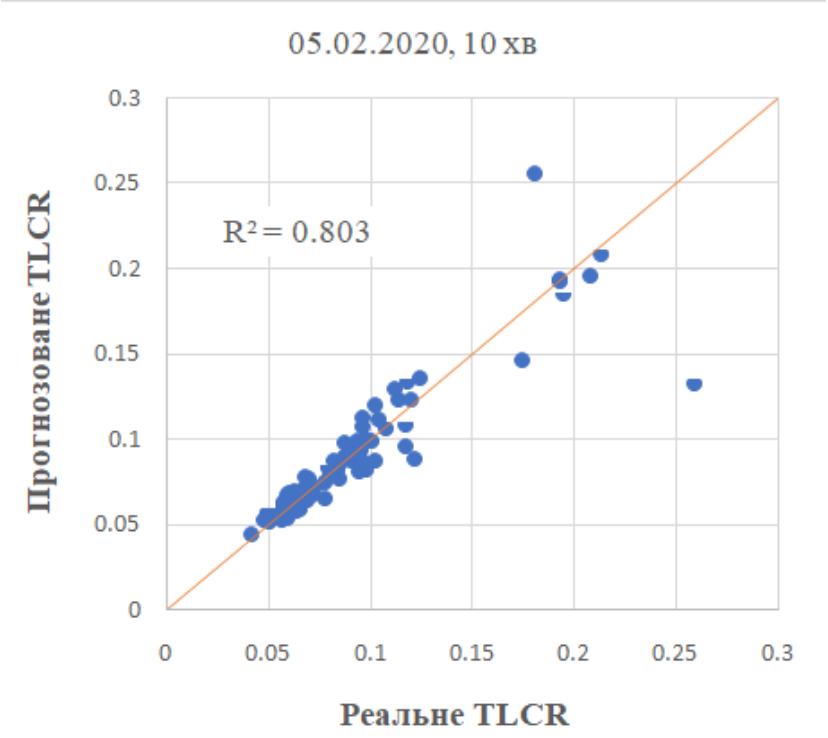


Рис. 10. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для середи з 10-ти хвилинним усередненням

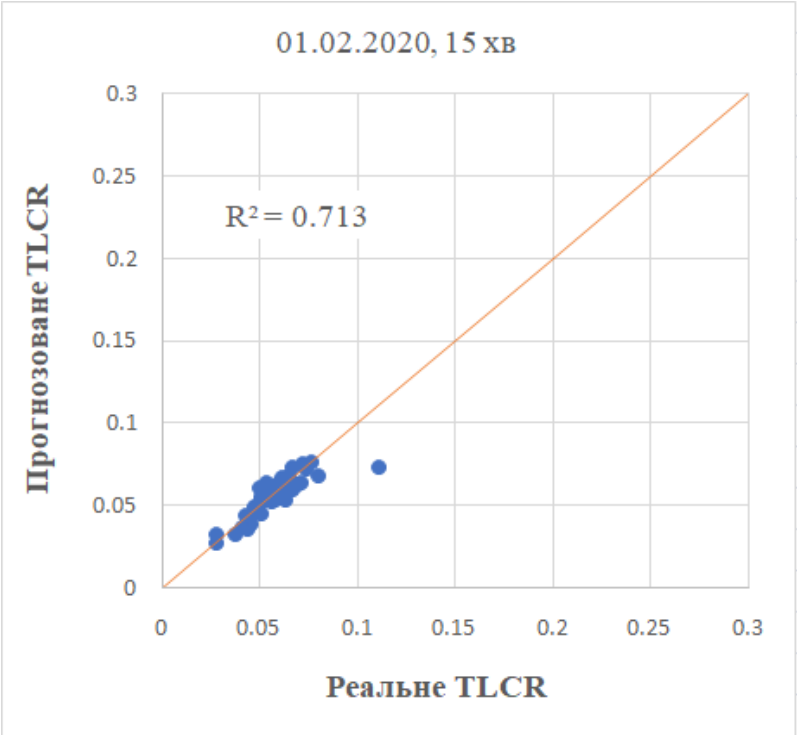


Рис. 11. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для середи з 10-ти хвилинним усередненням

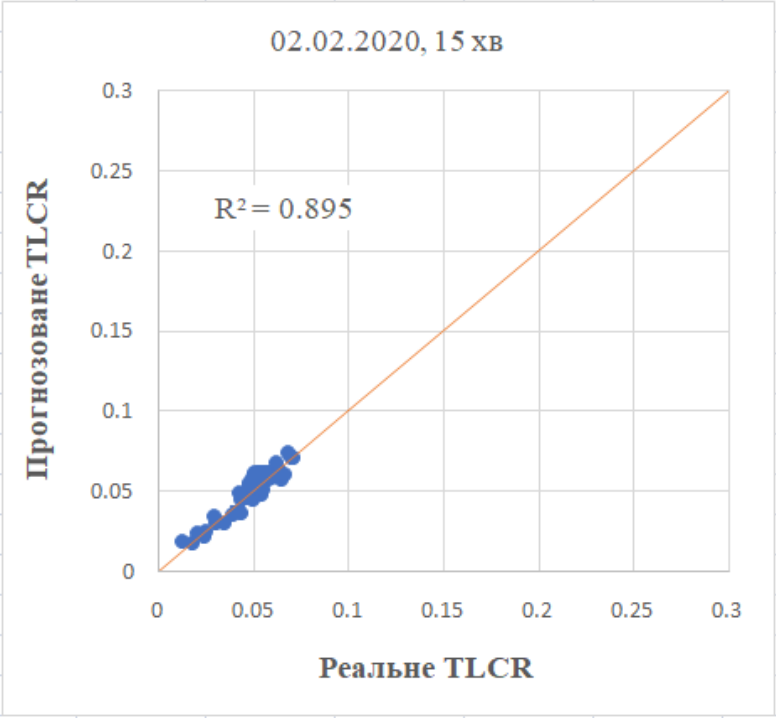


Рис. 12. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для неділі з 15-ти хвилинним усередненням

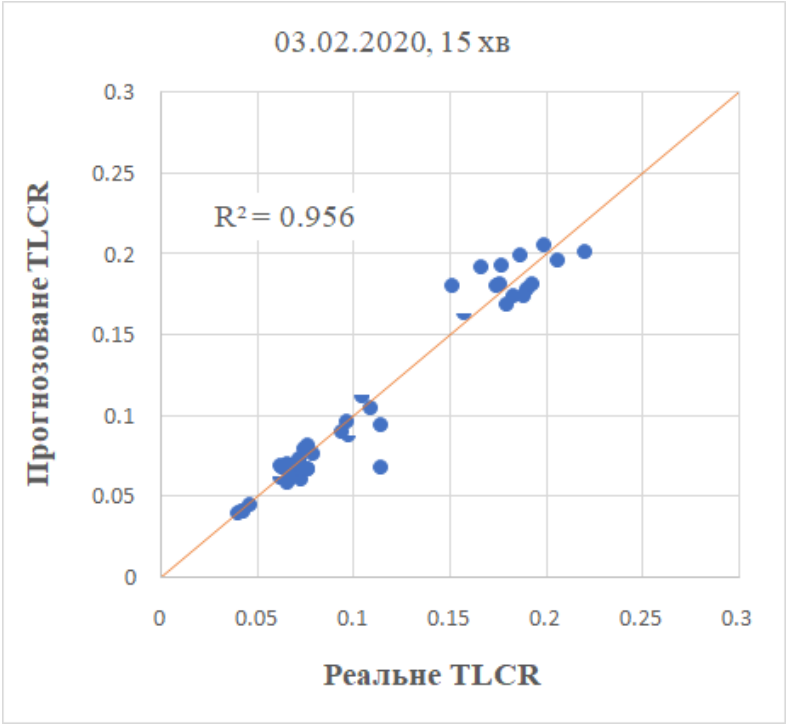


Рис. 13. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для понеділка з 15-ти хвилинним усередненням

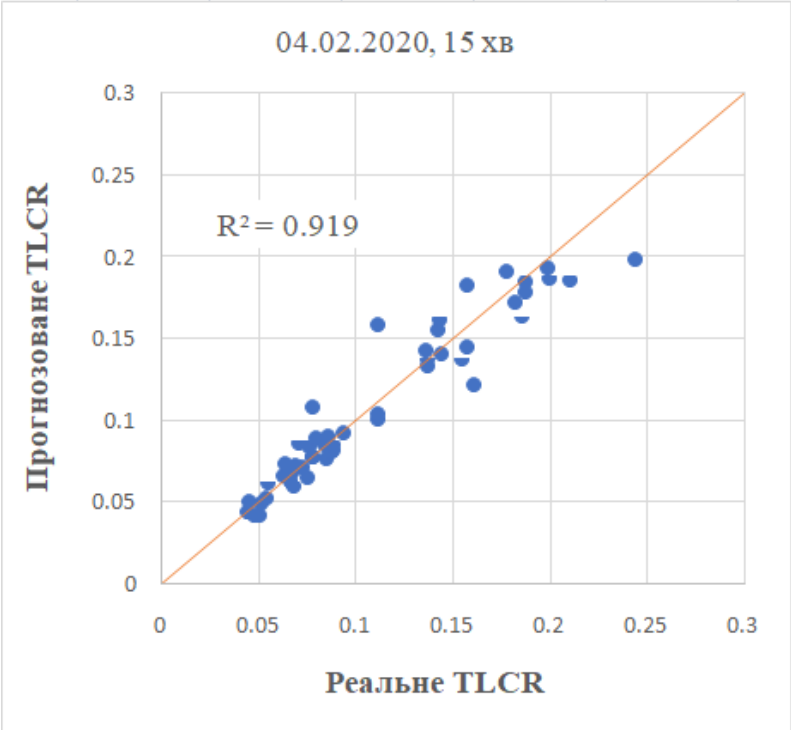


Рис. 14. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для вівторка з 15-ти хвилинним усередненням

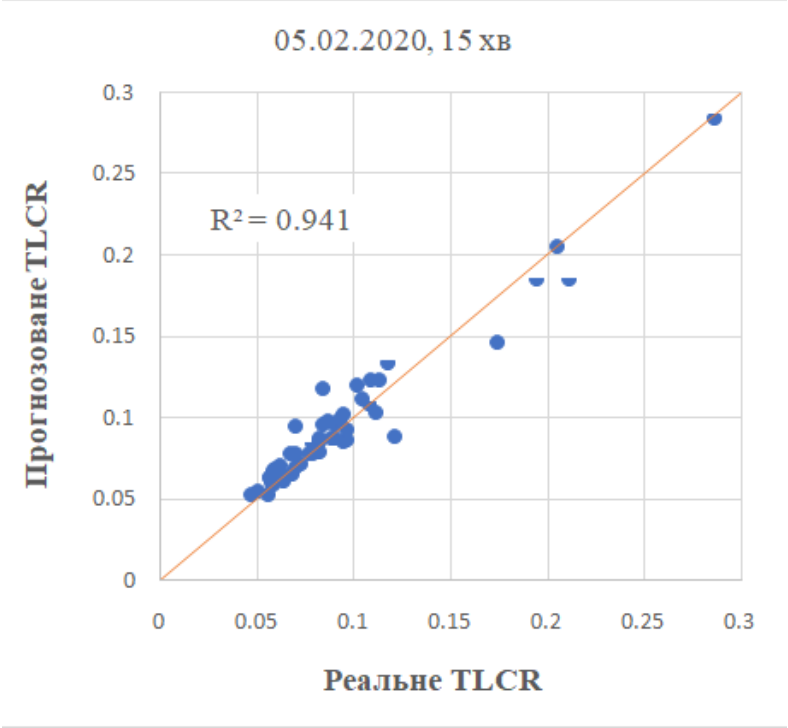


Рис. 15. Порівняння між прогнозованими значеннями потоку руху та реальними значеннями для середи з 15-ти хвилинним усередненням